Group 43

Final Project Report

CPSC 471

Jared Lundy, Jordan Lundy, Yuhao Guang

Abstract

Most of the existing medical database system we have today is disorganized and complex, therefore we aim to build a user-friendly and well-organized medical database system which might help billions of people worldwide to better store and manage their own medical records on a website-based database. The database we designed is not just the normal single-user database, as it allows four types of users (ie. doctor, nurse, patient and pharmacist) to login to their own account and will display different functionalities accordingly. It allows each user to view, input or edit the information stored in the existing database and the website is designed to be user-friendly and easy to use for all ages. Also, for the medical system database we implemented, it's reasonable that some new users would not have their login information in the database, therefore we created a register link below the front page which allows new user to create their own login information and will be added to the current users' database. Lastly, our medical system is always updated and allows all users to work together and refine the information, for example, if a doctor requests a certain drug at his end, after he inputs the name of the drug into the system, the pharmacist could fill the rest of the information needed (ie. company name of the drug, side effect, dosage etc.) at their end. This is just one of a plethora of functionalities that we will discuss in this report. Therefore, with the system we built, we're looking forward to contributing to either the provincial health care system or universal health care system for not only here in Canada, but anywhere around the world.

Introduction

Our database was designed to tackle the widespread issue of existing inefficient and complex medical systems. There are many existing online medical systems, however they are not very popular due to glaring issues and crippling issues within the functionalities. Therefore, it is necessary to create a database that makes it easy and efficient for every entity of a medical system to interact, as well as keep track of valuable information. Such a medical system should also be user friendly and minimalistic to ensure it can be used with ease by a plethora of different people with different levels of understanding of technology. Moreover, a glaring challenge in the medical community is the massive amounts of information. In many countries, including first world countries, this data has yet to be implemented in the form of medical data entries. It is typically kept as paper copies. This means that data can easily be lost, and isn't easily transferred to other medical professionals. Therefore, all medical information should gradually be stored online in a medical database.Our created database was therefore created to

fix the inadequacies and enhance the future of medical information storage. Our database therefore should help automate the medical field, as well as speed up slow processes, such as appointments and filling up medications. Our database should therefore save lives, which is the underlying importance of our medical system.

System Description

Our system, a medical database, creates a user friendly, and well connected system between many entities commonly found in a hospital/clinic. This system allows important data for each user to be stored, and retrieved easily. Also, this data can be transferred and viewed by other entities allowing ease of communication. The entities included are doctors, nurses, patients, and pharmacists. These are the main entities in a medical setting who would commonly need to communicate information, and were therefore deemed essential in our system. Our system keeps track of appointments, entity personal information, medical information, drugs, schedules, as well as connects the various entities when applicable. Also, our system includes various helpful functions, such as patient ID lookup, which further creates a user friendly experience, and easily gives access to valuable information to specific entities. There are also various view functions in order to allow certain entities to view important information, and sometimes make changes to information when necessary. In our system, we used drop down menus for ease of use. This was used where a set of valid inputs were determined, and therefore the user could only select from the given options. Next, we ensured all input forms were filled out properly again to ensure proper usage of our system. Our medical system stores the data entries in specific tables to ensure that data was well stored and therefore organized in an efficient manner as well. This increases a user-friendly environment, as well as helps to store medical information online, which is one of the various issues present in modern medical systems. Moreover, we split up the access between information in our data tables to ensure information was accessible to certain entities, and to ensure private data stayed private.

Project Design

One of the users for our system is the Nurse user. The role of nurses, as defined by our system, is a person that works at a clinic and is related to appointments at clinics. A nurse has the functionality of login/logout/registering (like every user), view schedule, update schedule, view clinics, assign clinics, search doctors, find their clinic and view their appointments. All of these functionalities ensure a thorough, well-defined nurse user who is able to do important actions

that a real-life nurse would have to do. The scheduling functionality for Nurse users includes updating and viewing schedules.The transactions involved in updating a schedule includes: inserting into the nurseschedule atuple with the schedule attributes that the nurse enters in the form (if they have no schedule yet), or updating the nurseschedule table for that nurse user (by key nurseID) and setting/updating the entered attributes to nurseschedule table in our database. This function is important to a nurse to ensure they can continuously have a reliable, dynamic schedule to dictate and plan their day to day affairs. View Schedule is of course just a select transaction to print out the schedule, and thus queries a select statement to the database to retrieve the table and print it out to the user. This functionality's importance to nurse is tied with updating schedule: our system giving nurses the ability to track their hectic schedules with various appointments and be able to have a reliable scheduling system. The clinic functionalities: View clinic, assign clinic, and get my clinic also provide various transactions. View clinic provides a select transaction (retrieval) from the nurse table to the database to print out all clinics to our webpage, and then gives a link option for each row (ie. each clinic) to assign that clinic. This link, if pressed, involves an update transaction to the nurse table, which will set the clinicname in this table to the selected clinic. Similarly, the assign clinic option (via the Nurse Homepage) allows for a nurse to update their clinicname table using a dropdown menu to select their clinic, and therefore is another update/set transaction for the nurse table. These functionalities are important to a nurse because a nurse needs to be able to be connected to the place that they work at. Assigning nurses is important for nurses to be able to freely move if they are transferred in real life to another clinic and it is critical that a transferred nurse can easily view information quickly on the clinic they move to. Finally, get my clinic provides a select transaction to transfer the clinic table (selected via clinicname matching the nurse's clinicname) from our database to the user website-end to easily view what clinic that nurse belongs to. Easy accesc to this information is important to a nurse, as a well-designed medical system should be able to quickly produce any clinic information for a nurse at their request. The next option via the nurse homepage, Search Doctors, is another select transaction (ie. a retrieval), which allows a nurse to select either all doctors or my clinic only and then see all doctors in the database (or only doctors in their clinic). This is a select retrieval transaction, and the 'all' or 'mycliniconly' option determines the restrictiveness of this transaction/select query. Since nurses are assistants to many different doctors at a time, it is crucial for a nurse user that they can quickly find a doctor and all their contact information quickly and reliably. Insufficiencies in this area severely impacts a nurse's ability to provide doctors with assistance, and therefore our system was built to prioritize this functionality for a nurse. It is also important that a nurse can filter all the doctors, especially by their clinic only, to view only applicable doctors without having to search through very large records. The last functionality provided to the nurse user by our web system is the view my appointments option accessed again via the nurse homepage. From the nurse-side of our system, this is again a select transaction that retrieves any appointment from the appointment table joined with the nurse table (by nurse id) where the foreign key of nurse id matches the id of the nurse logged in. As such, our system retrieves the table from our database with information of the nurse's assigned appointments and displays it for the nurse to easily view. This is again a critical action for a nurse, as their job depends on them being able to track appointments and provide care to every scheduled appointment. Unreliability here would be detrimental to a nurse and an entire medical unit, as appointments would be missed, and

therefore this is arguably the most important functionality of a nurse provided by our system. In total, a nurse has scheduling functionality, clinic functionality, appointment and doctor functionalities. This summarizes the functionality of the nurse user as well as the transaction collection of the nurse user on the back-end. For a step by step walkthrough (with snapshots of the webpage and database changes), seek the Nurse section of the user manual.

Another user in our system is the patient user. To fully describe the robust functionality and transaction collection provided to a patient we will go through each option separately. First, a patient has the logout/login/registration functionality, which obviously involves selecting transactions from the database to login and inserting to the patient table for registration (with selection to determine if it's a unique user). A patient in our system was designed to mirror a person in a real-world clinical setting. We wanted a patient to be able to do anything on our web system that a real-life patient would be able to do, and give them a reliable and easily accessible system to do this. A patient, as a result, has appointment, drug, clinic and doctor functionalities with corresponding database transactions for these functionalities. As a result, our system provides patients with a reliable system to book and track their medical history, keeping them involved and informed with their most important health decisions. Specifically, a patient is able to: search drug side effects, book and cancel appointments, view clinics, find a doctor, view their prescriptions and view their appointments. The first functionality, side effect search, provides a patient with the ability to enter any side effect and see any drugs that have this side effect. The transactions involved here is a retrieval, where we select tuples from the drug table in our database that have side-effects like what the patient entered, and return these applicable tuples to the webpage for the user to view. This functionality and transaction is very important to the health and well-being of a patient, who should always be well-informed of any side effects of their drugs and be able to check if one of their symptoms is caused by their drug(s). The next functionality for a patient is to book an appointment. This allows patients to make new appointments with doctors at clinics to receive medical care. This is one of the most important options for a patient, as this ensures that a patient can receive medical care via bookings, and connects nurses, doctors and patients in an ‘appointment’ and ‘has’ tuple instance in our database. When booking, a patient must enter the appointment type (eg checkup) and then select a clinic from the available clinics and from there they select a doctor (who works at that clinic) and set up the time and date. The transaction on the system end makes an appointment tuple with the entered information and links it to the patient via the patient’s ID, and creates a unique appointment number to assign to this inserted appointment tuple. Then, the system will create a tuple in the ‘has’ table that is linked to the appointment by the appointment number. This ‘has’ tuple will have the doctor information to connect the requested doctor, and will also contain time and date and patient information. Also, the system will assign an available nurse to the appointment and put their ID into the has tuple, as long as they work at that clinic. Thus, the entire transaction links the appointment and has tables, and inserts doctors, patients, nurses and other attribute information into each table to create a well-defined appointment. This appointment booking is critical for the medical care of the patient, and is also especially critical for the organization of the clinic that uses this system. Next, a patient has the ability to view clinics. This transaction is a simple retrieval from the database to the webpage, showcasing all attribute information to the patient. This is very useful

for patients to be able to easily contact prospective clinics. The next functionality for patient, doctor finder, is important for patients because it provides them the option to contact any doctor, which is critical to the medical care of a patient. Easy contact information for a doctor could save a patient's life in extreme circumstances, and thus this is very crucial for a patient to be able to do. In our doctor finder functionality, the user can view all doctors filtered by the clinic they work at. This transaction is a filtered retrieval from our database to our webpage, using a select,where condition to appropriately find the requested doctors at that clinic. Transferring the transaction result to the webpage gives a user-friendly and aesthetic table of all doctors filtered by clinic. The next functionality for patient is the view active prescriptions option. Tracking prescriptions is a chore for many patients, and can be incredibly difficult for elderly patients. Having an easy table of all their prescriptions with notes, dosage and other medical information is incredibly useful to patients, and ensures patients do not have to worry about missing doses or misusing their drugs. This function retrieves all drugs that the patient takes and shows all applicable drug information for each drug in a formatted table. The transactions needed to do this include a select query/retrieval from the prescribes table of our database, which contains the storage of a prescription and its patient when a doctor makes the prescription. For the patient, all we have to do transaction-wise is retrieve the tuple with a matching patient ID and return this row to the webpage. The final patient option is two-fold: view your appointments with an embedded cancel appointment option. The basic function of this option is to show the user all their booked appointments and all the necessary appointment information that is useful to a patient, with the option to cancel this appointment. Viewing appointments and being able to cancel them at the click of a button is incredibly useful for a patient. The transaction needed to do this includes a select from appointment joined with has that matches the patient id, which retrieves the necessary information for that appointment and brings it back to the patient user. If the cancel button is clicked, a delete transaction occurs, cascading a delete from the has and appointment tables for that user. This summarizes the patient functionality and the transaction collection for them. To view a complete walkthrough, refer to the patient section of the user manual.

Another user in our system is the pharmacist. The first functionality of a pharmacist, like any other entity in our system, is the login/register function. This allows an existing pharmacist entity to enter past the home page, to the list of other functionalities available to them. This means that that pharmacist would be within the pharmacist table in our database prior to this point. For a new pharmacist, they can register, which will ask for all the attributes required for the pharmacist table in our database. Then after this, if you were to look at the pharmacist table in our database, you would now see that newly added pharmacist as a new tuple. A pharmacist is an essential entity in our system. They are mainly in charge of the drugs in our system. Therefore, they can look at the list of drugs added into our database, order a new drug that isn't yet in our database, see the entire drug inventory in our database, and manage the existing drugs. Overall, they are the entity that deals with managing drugs, especially the information once new drugs are inputted into our system. Furthermore, they will have to order drugs when the inventory is low. When new drugs arrive, they then manage the information. This is similar to the role of a real world pharmacist. For their first functionality, which is looking at drugs in the system, they have the drug search function. In this, they can either search a drug by name, or

search drugs by company. This means that this drug search will go into the database and retrieve any drugs with the matching name, or the matching company. This means the information will be retrieved from the drugs table. This functionality is essential for the pharmacist so they can see if a certain drug is in the inventory. This allows pharmacists to ensure certain drugs, or company drugs are stocked. Next, they have the order a drug functionality. This, as stated before, is when they see a drug is missing or low in the inventory, and therefore the pharmacist decides the system needs more, just like a real pharmacist. In this functionality, they fill out a form that asks for the drug information that they are about to order, and when they submit it, then once the drug is delivered, it gets inserted into the database. Therefore, an insert will be made into the drugs table with all the applicable information. The next functionality is the view drug inventory. This, as stated before, allows the pharmacist to see the list of all the drugs in the inventory, rather than just a specific drug from before. Therefore, all the tuples in the drugs table will be retrieved and shown to the user. All the corresponding drug information will also be retrieved from this table and displayed. The pharmacist can also manage requested drugs. In our system, since the pharmacist is responsible for handling drugs and its corresponding information, this deals with the scenario that a doctor orders a drug (which they can only do so by name). In this situation, a dropdown menu will show all the newly added drugs that were recently ordered by a doctor. In this case, these drugs will be in the drugs table, but with NULL values for everything besides drug name. A pharmacist, like many cases in real life, knows more about the drug information, and therefore has to fill in the null values, which are the side effects and the company name. This means the previously added tuple will be updated, and the NULL values will be replaced with the pharmacist input. This functionality further upholds what a pharmacist's duties are, which was previously stated. A pharmacist, like any other entity, can also logout. This functionality simply logs them out, and doesn't actually alter or do anything with the database. For a step-by-step walk through for a pharmacist, refer to the pharmacist section of the user manual.

The last user for this system is the doctor user. The first functionality is logout.This doesn't alter the database, but only allows the current user to leave the session. Next, the doctor has the Get Schedule function. This is important since a doctor will have many appointments, and so they can keep track of all of them. This transaction simply retrieves the doctor's schedule from the database with all of its attributes. The Update Schedule function allows the doctor to alter their schedule by entering attributes. You can enter hours, days , and vacation days. This update transaction will update that doctor's schedule tuple in the doctor schedule table. The View Available Drugs allows the doctor to see what drugs are currently in the inventory. This allows a doctor to see what drugs they may prescribe to a patient. They press this button, and then a table of all available names are retrieved from the database drugs table, as well as a link to view drug info, which brings you a table with that drugs information, again from thge drugs table from the database. Therefore, this is a retrieval transaction with the option to do another retrieval from the database. Get Drug Info/Drug Search is a function that allows a doctor to retrieve drug information so they can be informed about the drugs they are prescribing. In this, a doctor would enter a drug name, which would be a retrieval transaction from the database, bringing in all the applicable drug information from drugs for that specific drug. Assign Nurses allows doctors to move nurses between clinics. Therefore, our system gives doctors the responsibility to control

the supply of nurses, and decide where they will work. In this, a doctor chooses a clinic, and chooses a nurse ID. This will be an update transaction that updates that nurse's clinic name in the nurse table of the database. The next functionality for doctor is the nurse ID lookup, which is provided as a convenience tool for doctors to quickly access the ID of nurses (useful for some forms). The webpage requires the entry of a first and last name of a nurse, and then the system does a select and retrieval transaction from the database (the nurse table where names match) showing the doctor on our website the numeric identifier for that searched nurse.The next functionality for doctor is the view prescriptions option. Here, a doctor can see any active prescriptions they have made to a specific person. To do so, they enter the first name and last name of the patient they want to see prescriptions for, and then the table of prescriptions and drug/prescription information is presented to the webpage. To do this, the system transaction includes a filtered select statement from prescribes that select prescribe tuples matching the logged in ID and matching the searched patient ID (which is searched by the system via selecting id from patient via the entered first and last name). This transaction then transfers the result to the webpage to be displayed. This transaction is important for a doctor to track any drug prescriptions they have made and to specific patients, safely and effectively managing drugs. Next is the prescribe drug option, which allows a doctor to make a prescription to a patient with a specific drug. A doctor needs to only select the drug to assign (based on a dropdown menu of available drugs), select the patient (with a patient dropdown of existing patients) and then fill in various drug information such as dosage, notes and date. This is important for doctors to actually give drugs to their patients and effectively treat them and provide medical prescription care. This is a very important job for doctors in real life, and is important for doctors in our system to be able to do. The transaction of the system inserts a prescribes tuple with the entered drug information by the doctor, and with the selected patient ID and drug name, which are foreign keys linked to the drug and the patient tables. Thus, this is a insert transaction on the doctor-end. The next functionality for doctor is the request new drug. This allows for the expansion of the drug inventory. Our system accomplishes this by allowing doctors to request drugs by name that are not available in the system yet, which adds the drug name to the system that a pharmacist has to fill and complete later. This is important because doctors need to have access to any drug to prescribe to patients, and this does not restrict the prescribing and treatment ability of a doctor. The transaction involved here is an insert into the drug table, with NULL values that a pharmacist manages to fulfill the request. A drug that is newly requested but not yet fulfilled by a pharmacist (ie. has nulls temporarily) can be prescribed but is limited due to its incompleteness (until pharmacists fill it in their portal). Next is the Insert Patient history, which allows a doctor to update patient history. This is important to maintain records for patients and preserve historical accuracy of these records. This is crucial to provide quality and accurate care to meet patient needs. Doctors need to enter the first name and last name of the patient (which the database searches in patient to retrieve their ID), and also enter history. The transaction involved is either an insert or an update to the patient_medicalhis table. Once the system selects the count of tuples in patient_medicalhis for that patient ID, if the count is zero it will insert a new tuple for that patient with the entered history. If the count is 1, it will update the history field by concatenating the history to maintain the history. Similarly, the insert patient conditions doctor option does the same thing as insert patient history (that we just discussed), except for the patient_conditions table instead. The next

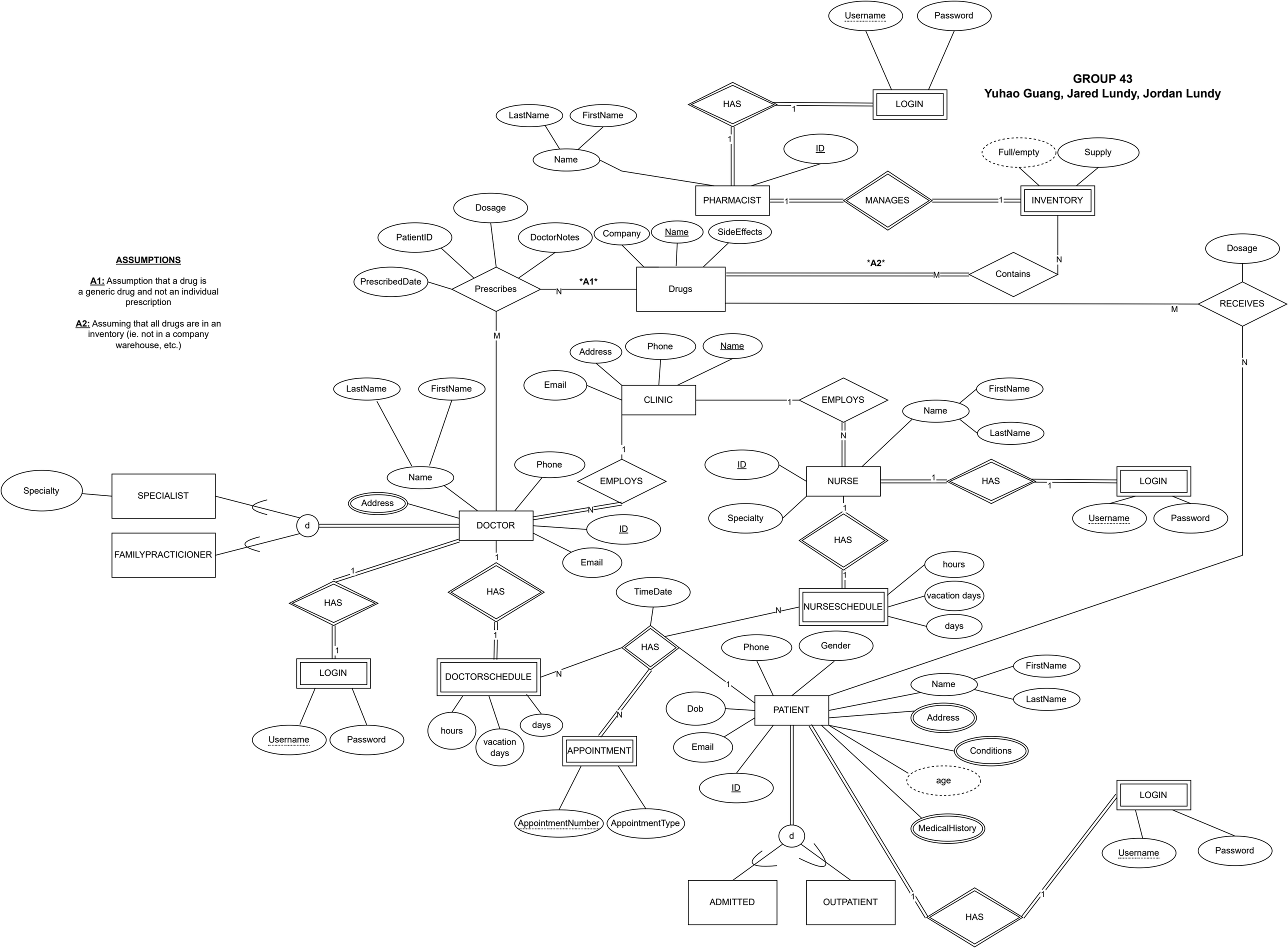
functionality is view patient history or conditions. This is important for doctors to actually see the medical history and conditions of the patients they treat, allowing them to actually provide care to their patients. To do this, the doctor enters the first and last name of the patient they are inquiring about. Then the select either conditions or history in the dropdown menu to see the patient medical history or their conditions. The transaction for this includes a select retrieval to obtain the patient ID from the entered first and last name (needed to index the patient_conditions or patient_medicalhis tables). Then, the system transaction does a select to transfer the result to the webpage to view. Next, the doctor has the option to view appointments. This is critical for time scheduling and planning of doctor users of our system. The transaction here is also a retrieval, indexing the ‘has’ table using the doctor ID. Finally, for convenience, a doctor has the option for patient ID lookup, where they enter a first name and last name for a patient and this searches the database for the ID of this person (using a select transaction). That concludes the entire doctor functionality and transaction collection for doctor. For a step by step walkthrough with snapshots and tracked database changes, see the doctor section of the user manual.

GROUP 43
Yuhao Guang, Jared Lundy, Jordan Lundy

ASSUMPTIONS

A1: Assumption that a drug is a generic drug and not an individual prescription

A2: Assuming that all drugs are in an inventory (ie. not in a company warehouse, etc.)



When implementing this project, the ERD diagram, the UML diagram and the sequence diagrams were considered at every step. Thus, no significant changes were made to the ERD, and the ERD was implemented in our system. The ERD was completely and accurately mapped to the relational model, which was then completely transferred into our mySQL database and our system. Because the relational model was completely implemented, we can confidently say the ERD that the relational model was mapped from was completely implemented. Therefore, the ERD was completely implemented in our system and is seen in both the tables in our database system (and attributes), as well as attributes in user java classes and the classes themselves.

Implementation Section

DRUGS

Name	SideEffects	Company

PHARMACIST

ID	Username	FirstName	LastName	Password	Supply

CONTAINS

Name

PRESCRIBES

ID	Name	Username	Prescribed	PatientID	DoctorNote	Dosage

DOCTOR

ID	Username	Password	Email	Phone	FirstName	LastName	Spe_flag	Specialty	Fam_flag	ClinicName

DOCTORSCHEDULE

ID	Hours	VacationDays	Days	Username

DOCTOR_ADDRESS

ID	Address	DoctorUser

RECEIVES

ID	Name	Username	Dosage

HAS

DoctorID	NurseID	PatientID	AppointmentNumber	PatientUser	DocUser	NurseUser	TimeDate

CLINIC

Name	Phone	Address	Email

NURSE

ID	Username	Password	Specialty	FirstName	LastName	ClinicName

NURSESCHEDULE

ID	Hours	VacationDa	Days	Username

PATIENT

ID	Username	Email	DOB	Phone	Gender	FirstName	LastName	Adm_flag	Out_flag	Password

APPOINTMENT

ID	AppointmentNumber	AppointmentType

PATIENT_ADDRESS

ID	Address

PATIENT_CONDITIONS

ID	Conditions

PATIENT_MEDICALHIS

ID	MedicalHistory

The relational model was completely integrated into our system. This is seen in our database in mySQL. Specifically, the tables match all relational model tables, as well as the keys, attributes and relationship connections. The schema as well for both the entire database and all tables match the relational model, and we did not deviate from the relational model. We did find, however, that some tables were not as useful in our actual system as others, such as the address tables. They were created and implemented into our system as planned, but were not heavily used in actuality.

The UML diagram was mostly followed, but some things were altered. We had to add some of the specific SQL methods that were not in there before, and we also found that some UML representations were not required (eg. stereotype classes were adequately reflected in the database itself and with joining tables these were not needed to have as a stereotype class in our code). The UML in this document has the added methods in the classes now.

The sequence diagram, however, was completely followed. Every method was implemented one-by-one based on the sequence diagram, and thus our system reflects this diagram completely.

PHARMACIST

FirstName: String
LastName: String
ID: int

assignInventory
changeInventory
updateInventory
changeLogin
retrieveInventory
getFirstName
setFirstName
getLastName
setLastName
getID
setID
getNullDrugs
dataDrugs
setUsername
getUsername
setPassword
getPassword
setSupply
getSupply
orderDrug

1..1 — 1..1

INVENTORY

Supply: String

addDrug
removeDrug
getInventory

1..1 — 1..*

DRUGS

Company: String
Name: String
SideEffects: String

addDrug
modifyDrug
addDoctor
changeDoctor
getDrugInfo
addPatient
getCompany
setCompany
getName
setName
getSideEffects
setSideEffects

RECEIVES

Dosage: String

0..* — 0..*

PATIENT

ID: int
Email: String
Dob: Date
Phone: String
Gender: {M,F, Nonbinary}
FirstName: String
LastName: String

setID
getID
setEmail
getEmail
calAge
setPhone
getPhone
setFirstName
getFirstName
setLastName
getLastName
deleteAppointment
viewApp
populateHas
getPatientID
pgetID
viewPrescriptions
bookAppointnment
setUsername
getUsername
setDOB
getDOB
setGender
getGender

0..* — 0..*

LOGIN_PATENT

Username: String
Password: String

password
update_password
getID

1..1 — 1..1

ADDRESS

Address: String

1..* — 0..*

CONDITIONS

Conditions: String[]

1..* — 0..*

MedicalHistory

History: String[]

1..* — 0..*

PRESCRIBES

prescribedDate: String
patientID: int
Dosage: String
DoctorNotes: String

0..* — 0..*

LOGIN_DOCTOR

Username: String
Password: String

password
update_password
getID

1..1 — 1..1

DOCTOR

FirstName: String
LastName: String
Phone: String
ID: int
Email: String

changeDoctor
assignDrugs
assignPatients
getSchedule
updateSchedule
updateLogin
assignClinic
assignNurses
addAddress
changeAddress
getFirstName
setFirstName
getLastName
setLastName
getPhone
setPhone
getID
setID
getEmail
setEmail
setUsername
getUsername
setPassword
getPassword
setSpecialty
getSpecialty
setCliniicName
getClinicName
insertDrug
insertHistory
insertConditions
viewAppointmentDoc
getConiditions
viewHist
getSchedule
updateSchedule
assignNurses
getDocID
viewPrescriptions
docPrescribes
findID

0..* — 1..*

ADDRESS

Address: String

DOCTORSCHEDULE

Hours: Integer
VacationDays: String[]
Days: String

updateHours
setVacday
getvacday
changeDoctor
getHours
setDays
getDays

0..1 — 1..1

1..* — 1..*

OUTPATIENT

Extends

ADMITTED

Extends

APPOINTMENT

AppointmentType
AppointmentNumber

setApptType
getApptType
setApptNum
getApptNum
assignDoc
assignNurse
assignPatient

0..1 — 0..1

TIMEDATE

TimeDate: String

CLINIC

Email: String
Address: String
Phone: String
Name: String

addDoctor
removeDoctor
addNurse
removeNurse
getEmail
getAddress
getPhone
getName
setEmail
setAddress
setPhone
setName

1 — 1..1

1 — 1..*

NurseSchedule

Hours: int
VacationDays: String []
Days: string

updateHours
setVacday
getVacday
changeNurese
updateDays
getHours
setDays
getDays

0..1 — 1..1

1..* — 0..1

FAMILY PRACTITIONER

Extends

SPECIALIST

Specialty

Extends

NURSE

ID: int
Specialty: String
FirstName: String
LastName: String

getSchedule
updateSchedule
assignClinic
assignSpecialty
getClinic
getSpecialty
setFirstName
getFirstName
setLastName
getLastName
getID
setID
viewAppointment
updateSchedule
setClinicNasme
seUsername
setPassword
setSpecialty
getUsername
getPassword
getSpeciaty
getClinicName

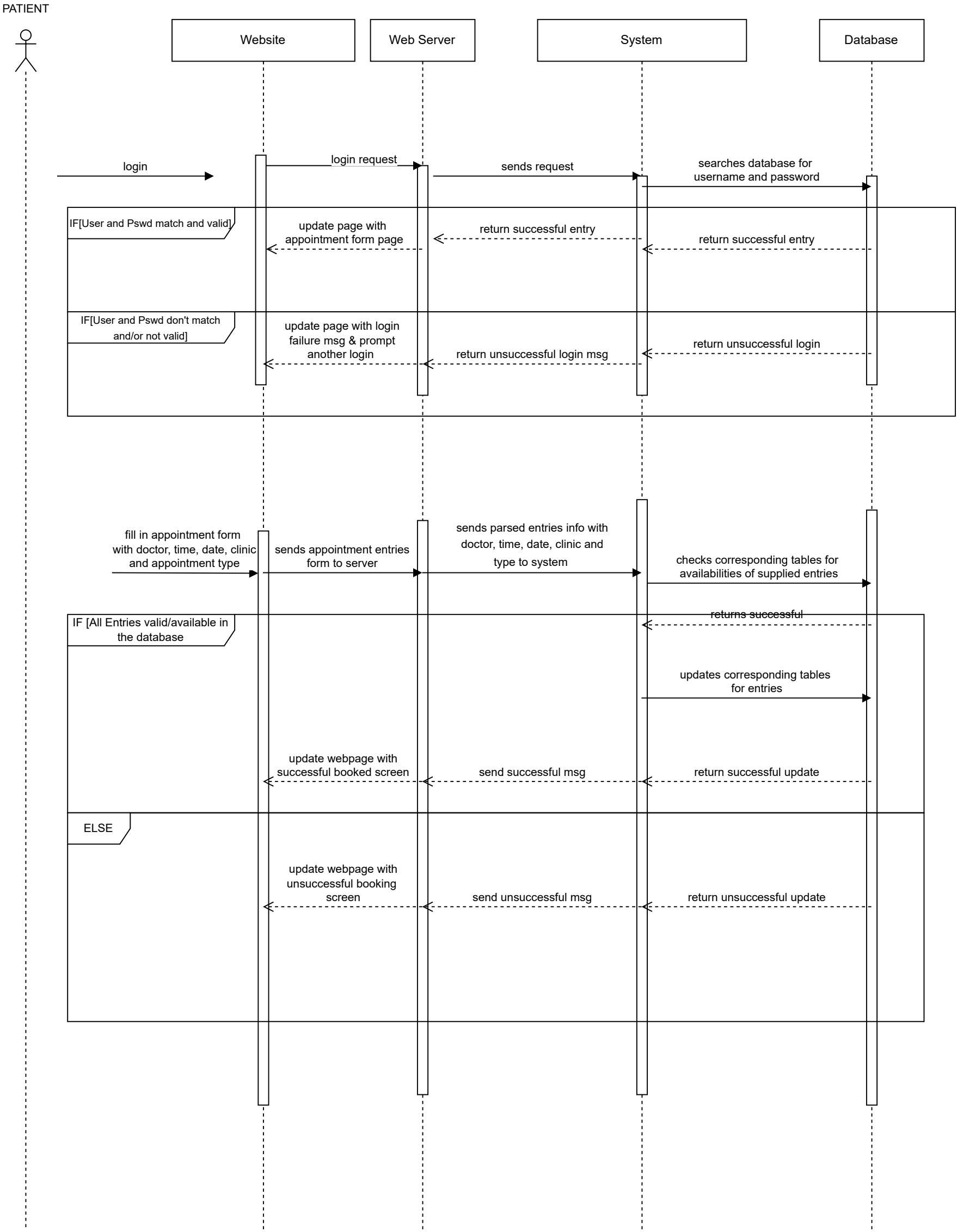
1..1 — 1..1

LOGIN_NURSE

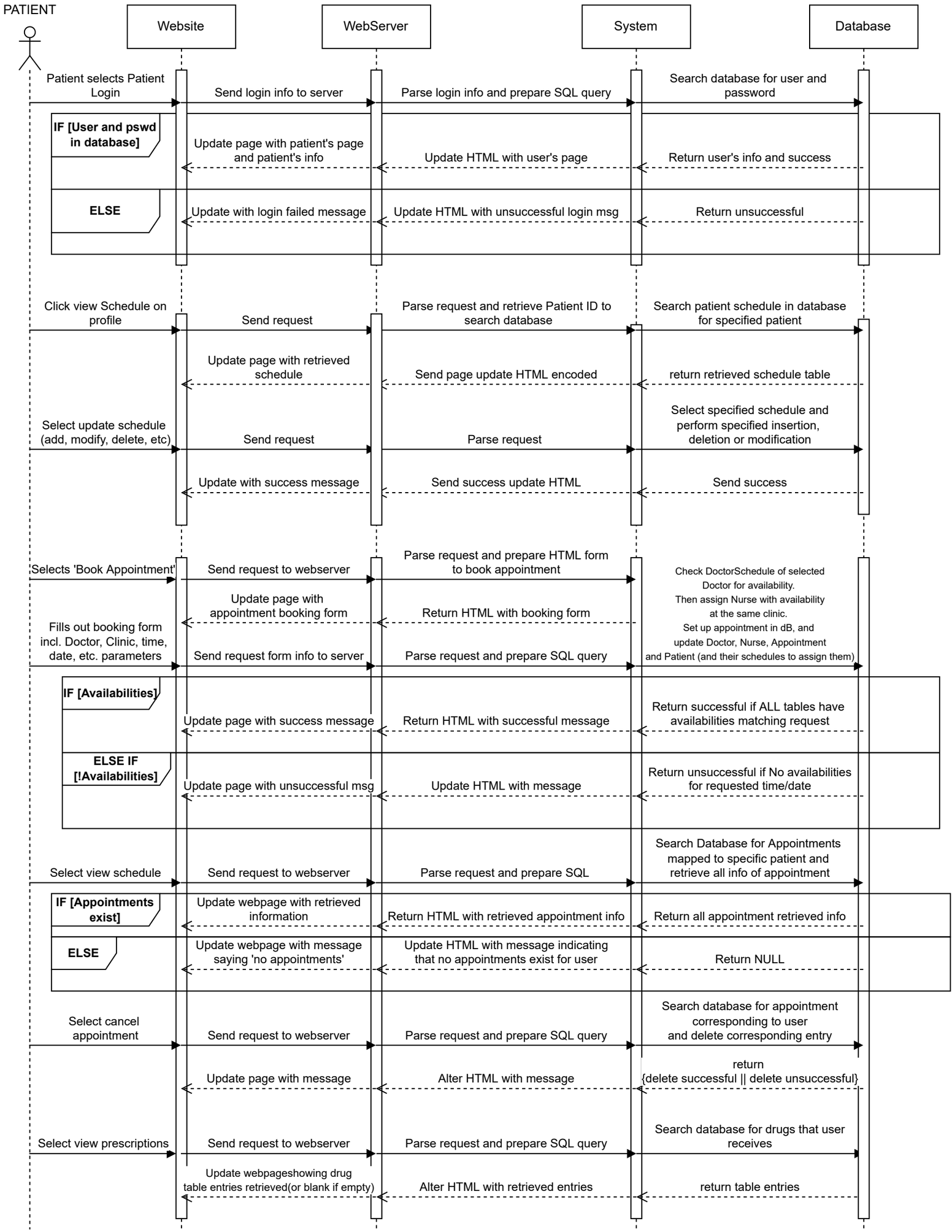
Username: String
Password: String

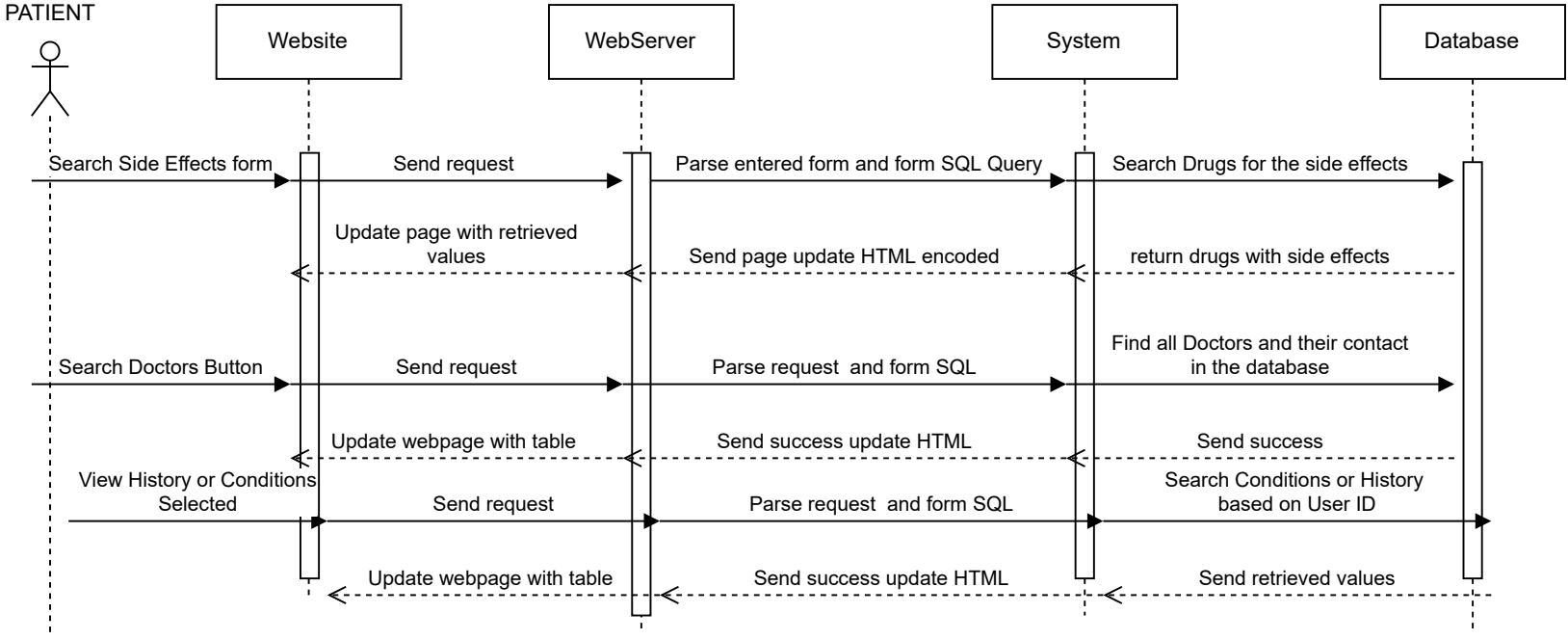
password
update_password
getID

USE-CASE: Patient Booking an Appointment

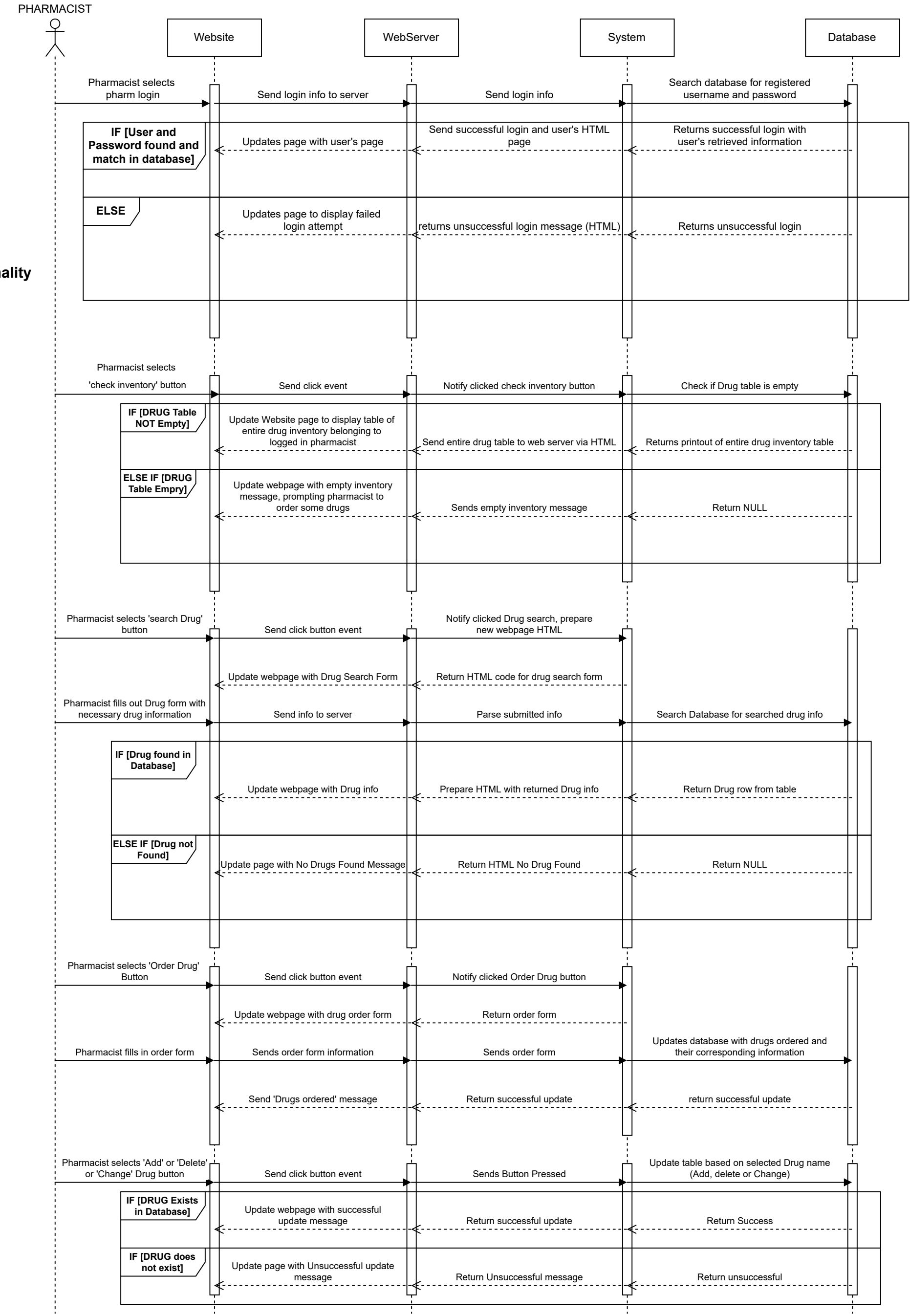


USE-CASE: PATIENT FUNCTIONALITY

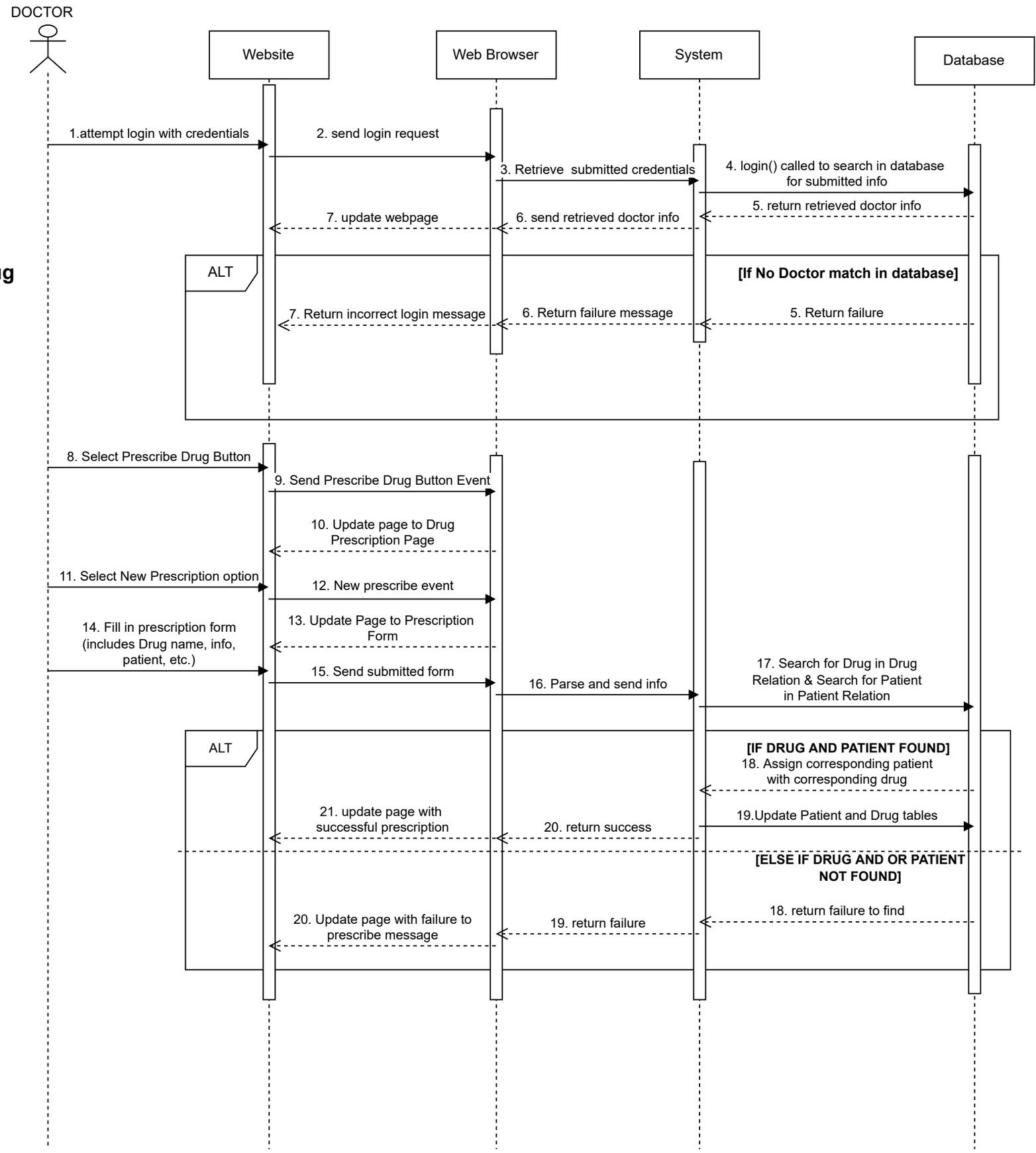


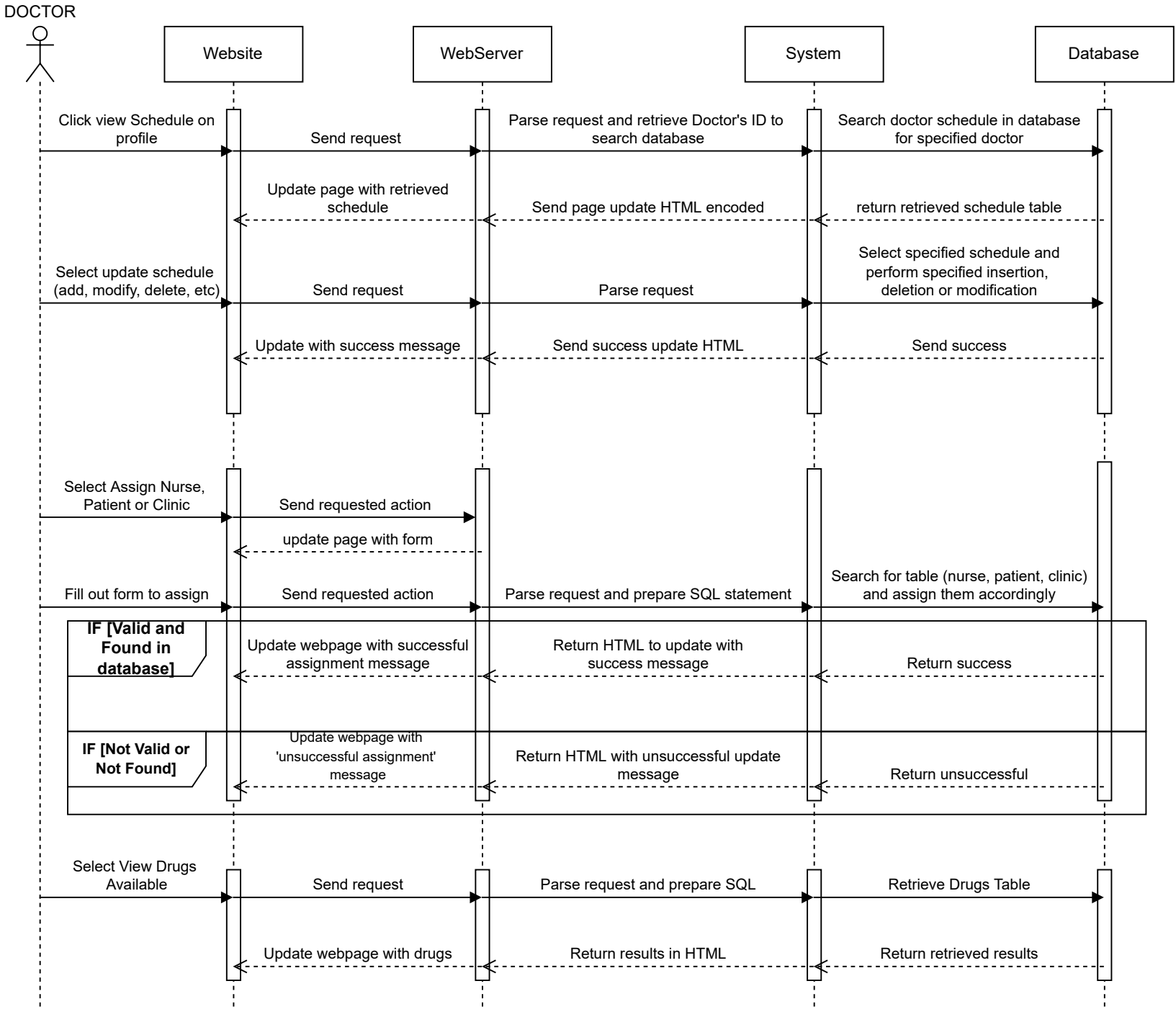

PATIENT
Website
WebServer
System
Database
Search Side Effects form
Send request
Parse entered form and form SQL Query
Search Drugs for the side effects
Update page with retrieved values
Send page update HTML encoded
return drugs with side effects
Search Doctors Button
Send request
Parse request and form SQL
Find all Doctors and their contact in the database
Update webpage with table
Send success update HTML
Send success
View History or Conditions Selected
Send request
Parse request and form SQL
Search Conditions or History based on User ID
Update webpage with table
Send success update HTML
Send retrieved values

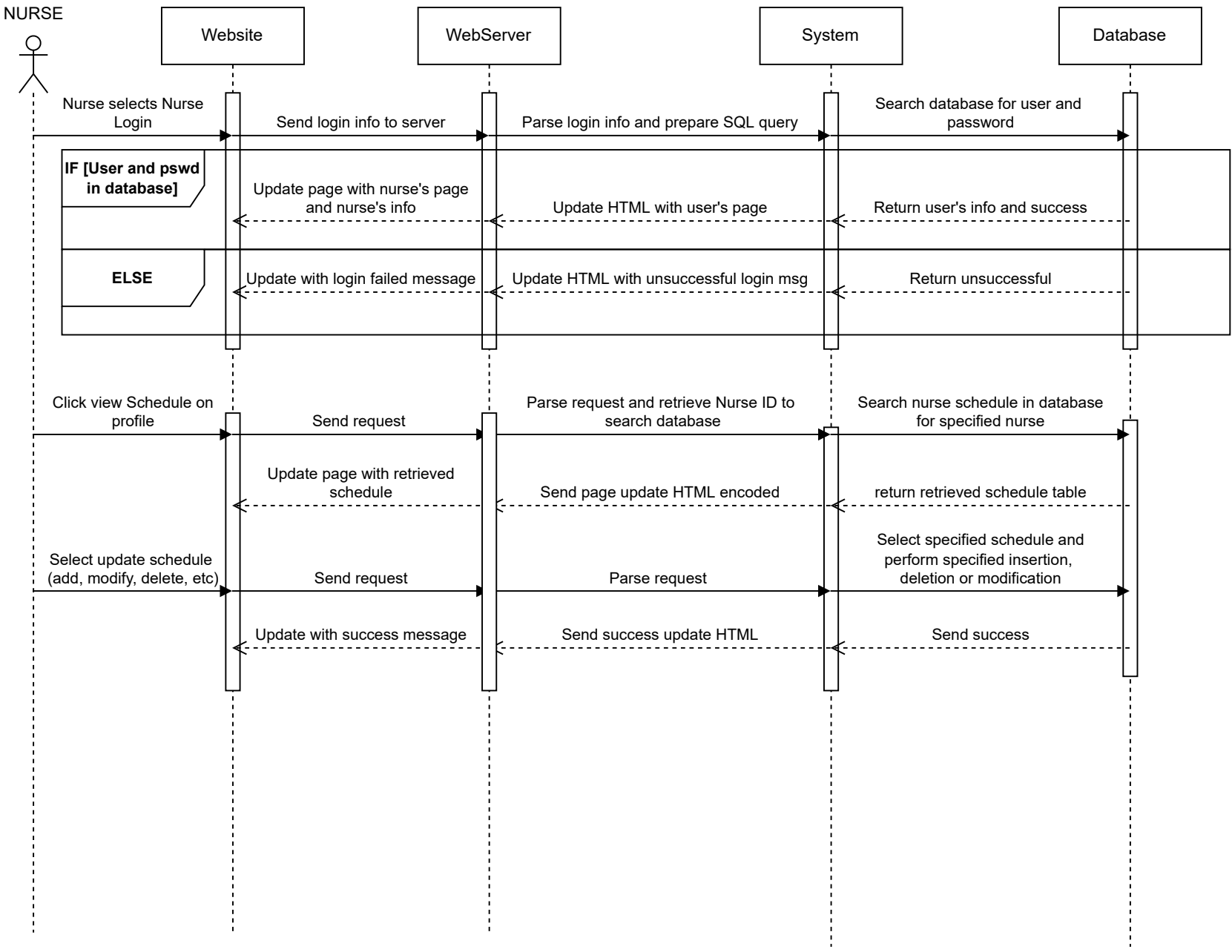
USE-CASE:
Pharmacist Functionality

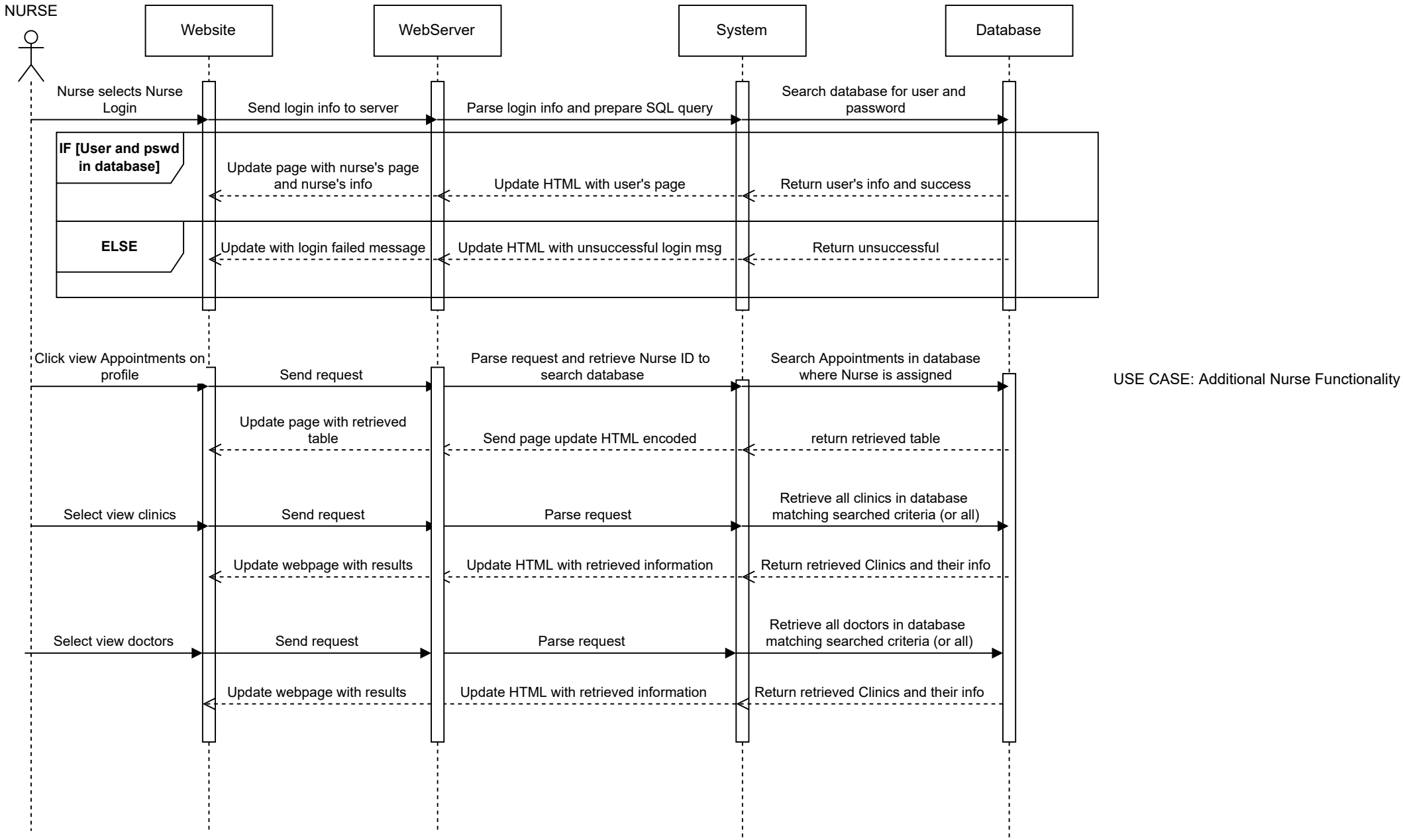


USE CASE: Doctor Prescribing Drug




DOCTOR
Website
WebServer
System
Database
Click view Schedule on profile
Send request
Parse request and retrieve Doctor's ID to search database
Search doctor schedule in database for specified doctor
Update page with retrieved schedule
Send page update HTML encoded
return retrieved schedule table
Select update schedule (add, modify, delete, etc)
Send request
Parse request
Select specified schedule and perform specified insertion, deletion or modification
Update with success message
Send success update HTML
Send success
Select Assign Nurse, Patient or Clinic
Send requested action
update page with form
Fill out form to assign
Send requested action
Parse request and prepare SQL statement
Search for table (nurse, patient, clinic) and assign them accordingly
IF [Valid and Found in database]
Update webpage with successful assignment message
Return HTML to update with success message
Return success
IF [Not Valid or Not Found]
Update webpage with 'unsuccessful assignment' message
Return HTML with unsuccessful update message
Return unsuccessful
Select View Drugs Available
Send request
Parse request and prepare SQL
Retrieve Drugs Table
Update webpage with drugs
Return results in HTML
Return retrieved results


NURSE
Website
WebServer
System
Database
Nurse selects Nurse Login
Send login info to server
Parse login info and prepare SQL query
Search database for user and password
IF [User and pswd in database]
Update page with nurse's page and nurse's info
Update HTML with user's page
Return user's info and success
ELSE
Update with login failed message
Update HTML with unsuccessful login msg
Return unsuccessful
Click view Schedule on profile
Send request
Parse request and retrieve Nurse ID to search database
Search nurse schedule in database for specified nurse
Update page with retrieved schedule
Send page update HTML encoded
return retrieved schedule table
Select update schedule (add, modify, delete, etc)
Send request
Parse request
Select specified schedule and perform specified insertion, deletion or modification
Update with success message
Send success update HTML
Send success


NURSE
Website
WebServer
System
Database
Nurse selects Nurse Login
Send login info to server
Parse login info and prepare SQL query
Search database for user and password
IF [User and pswd in database]
Update page with nurse's page and nurse's info
Update HTML with user's page
Return user's info and success
ELSE
Update with login failed message
Update HTML with unsuccessful login msg
Return unsuccessful
Click view Appointments on profile
Send request
Parse request and retrieve Nurse ID to search database
Search Appointments in database where Nurse is assigned
Update page with retrieved table
Send page update HTML encoded
return retrieved table
Select view clinics
Send request
Parse request
Retrieve all clinics in database matching searched criteria (or all)
Update webpage with results
Update HTML with retrieved information
Return retrieved Clinics and their info
Select view doctors
Send request
Parse request
Retrieve all doctors in database matching searched criteria (or all)
Update webpage with results
Update HTML with retrieved information
Return retrieved Clinics and their info
USE CASE: Additional Nurse Functionality

We used MySQL as our DBMS. We created all our tables for our system using MySQL. We created our database using MySQL. We then used all of our SQL queries to control, store, and alter our data. In order to code using MySQL, we used Java. We used the Java JDBC to connect to our MySQL database. This allowed us to make SQL queries as strings, and then execute them to our database through the JDBC driver.
The basic flow of our system is described as such:
- System homepage html is the default beginning page (when system ran on server first)
- Buttons to login and register send you to login or registration html forms, which upon submission sends the data submitted to java servlets, which then call user java files (eg. doctor.java, etc) methods that connect to the database and execute queries for logging in and registering and sends results back to the servlet Importantly, the servlet sets session attributes for username, first name, last name, etc. for the currently logged in user that can be used for every other file and function (provides dynamic ID)
- If valid, the servlet redirects to the user dynamic homepage (eg. doctor home) for that user (a jsp file), if invalid redirects back to login (with html message)
- the jsp contains session attributes and links to every function/option that that user can do
- clicking any of these links forwards to the corresponding html form to enter attributes, then the html file sends the data to the get method of the corresponding java servlet. The servlet parses the data and obtains the user's session attributes and calls the user java method (eg. method in doctor.java to find drugs). This method will connect to the database and execute query(ies) based on the method functionality. The result is sent back to the servlet, which then redirects the result as html and redirects to the corresponding html or jsp page.
- At any point, there is an html a href link to the user's homepage to cancel any option
- Any result and message is now displayed on the html page, and links allow for the user to go back to their homepage and press another function/option
- Users can logout, which brings the user back to the system homepage and now any other user can login and do the same things (dynamic for user and user type)

Therefore, the basic code flow of our system is:
(Login/register: HTML system homepage -> HTML form -> servlet -> java method + SQL connection ->servlet -> jsp user homepage)
(once logged in: JSP User Home -> HTML form -> java servlet -> java method + SQL connection -> servlet -> corresponding JSP or HTML page (with link back to homepage))

This holds true for almost every case, although some steps do not need an html form and go directly from HTML to servlet with session parameters.

The following SQL statements were used in our project to implement all transactions:
(*In case of any missing SQL statements below, see the source code submission*)

Doctor SQL statements:

```
Requesting and inserting a new drug:
String q="INSERT INTO Drugs Values(NULL, NULL, ?)";

Selecting clinic name:
String query="SELECT Name FROM CLINIC";



Selecting Nurse IDs:
String query2="SELECT ID FROM NURSE";

Obtaining count of doctors with a username:
String query="SELECT COUNT(*) FROM DOCTOR WHERE username=?";

Selecting doctor info with logged in username:
String query2="SELECT * FROM DOCTOR WHERE username=?";

Selecting drug names:
String query="SELECT Name FROM DRUGS";

Selecting patient ids:
String query2="SELECT ID FROM PATIENT";

Selecting drug information:
String query="SELECT * FROM DRUGS";

Getting count of patient medical history entries
String str="SELECT COUNT(*) FROM PATIENT_MEDICALHIS WHERE ID=?";

Getting patient medical history
String s="SELECT MedicalHistory FROM PATIENT_MEDICALHIS WHERE ID=?";

Updating a patient’s medical history
s="UPDATE PATIENT_MEDICALHIS SET MedicalHistory=? WHERE ID=?";

Inserting into medical history
String ss="INSERT INTO PATIENT_MEDICALHIS VALUES(?,?)";


Getting a count of a patient’s conditions
String str="SELECT COUNT(*) FROM PATIENT_CONDITIONS WHERE ID=?";

Getting a patient’s medical conditions
String s="SELECT Conditions FROM PATIENT_CONDITIONS WHERE ID=?";

Updating a patient’s medical conditions
s="UPDATE PATIENT_CONDITIONS SET Conditions=? WHERE ID=?";

Inserting a patient’s conditions
String ss="INSERT INTO PATIENT_CONDITIONS VALUES(?,?)";

Getting appointment info
String q="SELECT HAS.AppointmentNumber, AppointmentType, TimeDate, Appointment.ID FROM
Appointment, HAS WHERE HAS.DoctorID=? AND
Appointment.AppointmentNumber=HAS.AppointmentNumber";
```

Getting a patient's conditions
```
String w="SELECT Conditions FROM PATIENT_CONDITIONS WHERE ID=?";
```

Getting medical history for a patient
```
String query="SELECT MedicalHistory FROM PATIENT_MEDICALHIS WHERE ID=?";
```

Getting a doctor's schedule
```
String q="SELECT Fname, Lname, ClinicName, Hours, VacatonDays, Days FROM doctor,
doctorschedule WHERE doctor.username=doctorschedule.username AND
doctorschedule.username=?";
```

Getting count of a doctor's scheudle
```
String check="SELECT COUNT(*) FROM doctorschedule WHERE username=?";
```

Getting a doctor ID
```
String get_id="SELECT ID FROM DOCTOR WHERE Username=?";
```

Inserting into doctor schedule
```
String balan="INSERT INTO doctorschedule VALUES(?, ?, ?, ?, ?)";
```

Updating a doctor's schedule
```
String q= "UPDATE doctorschedule SET hours=?, vacatondays=?, days=? WHERE Username=?";
```

Updating nurse info
```
String q="UPDATE nurse SET ClinicName=? WHERE ID=?";
```

Getting a doctor's ID
```
String s="SELECT ID FROM doctor WHERE username=?";
```

Viewing a prescription a doctor gave out
```
String qu="SELECT Name,Prescribed,DoctorNotes,Dosage FROM prescribes WHERE PatientID=?
AND username=?";
```

inserting/prescribing new prescription
```
String q="INSERT INTO prescribes VALUES(?,?,?,?,?,?,?)";
```

Getting a nurse ID
```
String query="SELECT ID FROM nurse WHERE Fname=? AND Lname=?";
```

Inserting newly registered doctor:
```
String query="INSERT INTO DOCTOR VALUES(?,?,?,?,?,?,?,?,?,?,?,?)";
```

Search drugs by name:
```
String query="SELECT * FROM DRUGS WHERE Name=?";
```

Pharmacist SQL Statements:

```
Search drugs by company name:
String query="SELECT * FROM DRUGS WHERE Company=?";

Search drugs by name:
String query="SELECT * FROM DRUGS WHERE Name=?";

Inserting registered nurse:
String query="INSERT INTO PHARMACIST VALUES(?,?,?,?,?,?)";

Obtaining count of registered pharmacists:
String query="SELECT COUNT(*) FROM PHARMACIST WHERE username=?";

Selecting logged in pharmacist's information:
String query2="SELECT * FROM PHARMACIST WHERE username=?";

Selecting drugs:
String query="SELECT * FROM DRUGS";

Inserting information into drugs table
query="INSERT INTO DRUGS VALUES(?,?,?)";

Getting all drugs with null information for company and side effects
String q="SELECT Name FROM DRUGS WHERE Company IS NULL AND SideEffects IS NULL";

Updating drugs with null information inserted by a doctor when ordering a new drug
String s="UPDATE DRUGS SET SideEffects=?, Company=? WHERE Name=?";
```

Patient SQL Statements:

```
Selecting doctor and clinic for an appointment:
String the_query="SELECT ID, Username FROM DOCTOR WHERE LName=? AND ClinicName=?";

System assigning random nurse to patient appointment (that works at clinic):
the_query="SELECT ID, Username FROM NURSE WHERE ClinicName=? ORDER BY RAND() LIMIT 1";

Retrieve username:
the_query="SELECT Username FROM PATIENT WHERE ID=?";

Retrieving logged in patient id:
String getid="SELECT ID FROM PATIENT WHERE USERNAME=?";

System calculating appointment number based on existing appointments:
String  getid2="SELECT MAX(AppointmentNumber) FROM APPOINTMENT";

Retrieve clinic:
```

```
String query="SELECT * FROM CLINIC";

Getting doctor lastname from requested clinic:
String the_query="SELECT Lname FROM DOCTOR WHERE ClinicName=?";

Getting clinic names:
String query="SELECT Name FROM CLINIC";

Finding number of patients with username:
String query="SELECT COUNT(*) FROM PATIENT WHERE username=?";

Retrieve patient info:
String query2="SELECT * FROM PATIENT WHERE username=?";

Selecting clinics:
String query="SELECT * FROM CLINIC";

Getting all patient's information as well as prescribes informationfrom the prescribes
table
String qu="SELECT Name,Prescribed,DoctorNotes,Dosage FROM prescribes WHERE
PatientID=?";

Searching doctor information by clinic:
String query = "SELECT Fname,Lname,Email,Specialty,ClinicName,Phone FROM DOCTOR WHERE
ClinicName=?";

Retrieve doctor information
String query = "SELECT Fname,Lname,Email,Specialty,ClinicName,Phone FROM DOCTOR";



 Finding drugs based on searched sideeffects:
 String query="SELECT * FROM DRUGS WHERE SideEffects LIKE ?";



Registering patient:
String query="INSERT INTO PATIENT VALUES(?,?,?,?,?,?,?,?,?,?)";

Inserting values into the HAS table
String str="INSERT INTO HAS VALUES(?,?,?,?,?,?,?,?)";

Deleting an appointment
String s="DELETE FROM Appointment WHERE ID=? AND AppointmentNumber=?";

Getting an appointment information
String q="SELECT HAS.AppointmentNumber, AppointmentType, TimeDate FROM Appointment,
HAS WHERE HAS.PatientID=? AND Appointment.AppointmentNumber=HAS.AppointmentNumber";


Getting a patient's ID
String str="SELECT ID FROM patient WHERE FName=? AND LName=?";

Getting a patient's ID that matches a username
```

```
String query="SELECT ID FROM patient WHERE username=?";

Getting count from appointment based on an ID
String query="SELECT COUNT(*) FROM Appointment WHERE ID=?";

Inserting an appointment
String qq="INSERT INTO Appointment VALUES(?,?,?)";
```

Nurse SQL

```
Selecting doctors that work in nurse's clinic:
query="SELECT D.Fname,D.Lname,D.Email,D.Specialty,D.ClinicName,D.Phone FROM DOCTOR AS
D,NURSE AS N WHERE D.ClinicName=N.ClinicName AND N.Username=?";

Checking how many clinics with a certain name
query="SELECT COUNT(*) FROM CLINIC WHERE Name=?";

Setting a nurse's clinic
query="UPDATE NURSE SET ClinicName=? WHERE Username=?";

Getting a nurse schedule
String q="SELECT Fname, Lname, ClinicName, Hours, VacatonDays, Days FROM nurse,
nurseschedule WHERE nurse.username=nurseschedule.username AND
nurseschedule.username=?";

Getting all nurse schedule's for a nurse
String check="SELECT COUNT(*) FROM nurseschedule WHERE username=?";

Getting an ID for a nurse
String get_id="SELECT ID FROM NURSE WHERE Username=?";

Putting in a new nurse schedule
String balan="INSERT INTO nurseschedule VALUES(?, ?, ?, ?, ?)";

Updating a nurse schedule
String q= "UPDATE nurseschedule SET hours=?, vacatondays=?, days=? WHERE Username=?";


String q="SELECT address, email, phone, name FROM nurse AS n JOIN clinic AS c ON
clinicname=name WHERE n.username=?";
```

Viewing appointments for a nurse

```
String q="SELECT HAS.AppointmentNumber, AppointmentType, TimeDate, Appointment.ID FROM
Appointment, HAS WHERE HAS.NurseUser=? AND
Appointment.AppointmentNumber=HAS.AppointmentNumber";
```

Retrieve number of nurses with username:

```
String query="SELECT COUNT(*) FROM NURSE WHERE username=?";
```

Get nurse info for logged in nurse:

```
String query2="SELECT * FROM NURSE WHERE username=?";
```

Selecting clinics:

```
String query="SELECT * FROM CLINIC";
```

Registering a patient:

```
String query="INSERT INTO PATIENT VALUES(?,?,?,?,?,?,?,?,?,?)";
```

Searching for doctors:

```
String query = "SELECT Fname,Lname,Email,Specialty,ClinicName,Phone FROM DOCTOR WHERE
ClinicName=?";
```

Find doctor information:

```
String query = "SELECT Fname,Lname,Email,Specialty,ClinicName,Phone FROM DOCTOR";
```

Getting doctors who work in same clinic as the nurse:

```
query="SELECT D.Fname,D.Lname,D.Email,D.Specialty,D.ClinicName,D.Phone FROM DOCTOR
AS D,NURSE AS N WHERE D.ClinicName=N.ClinicName AND N.Username=?";
```

User Manual

This user manual will go through the entire functionality of our built website, including every functionality and option for each user type of our system. This manual will provide snapshots for every functionality as well. The underlying MySQL Database transactions can be viewed in the appendix section, and will be referred to as we go along. Note that the figures are labelled *figure U.x,* where U denotes this image belongs to the user manual section, and x is the number of the image.

The first discussion of our webpage starts at the homepage:

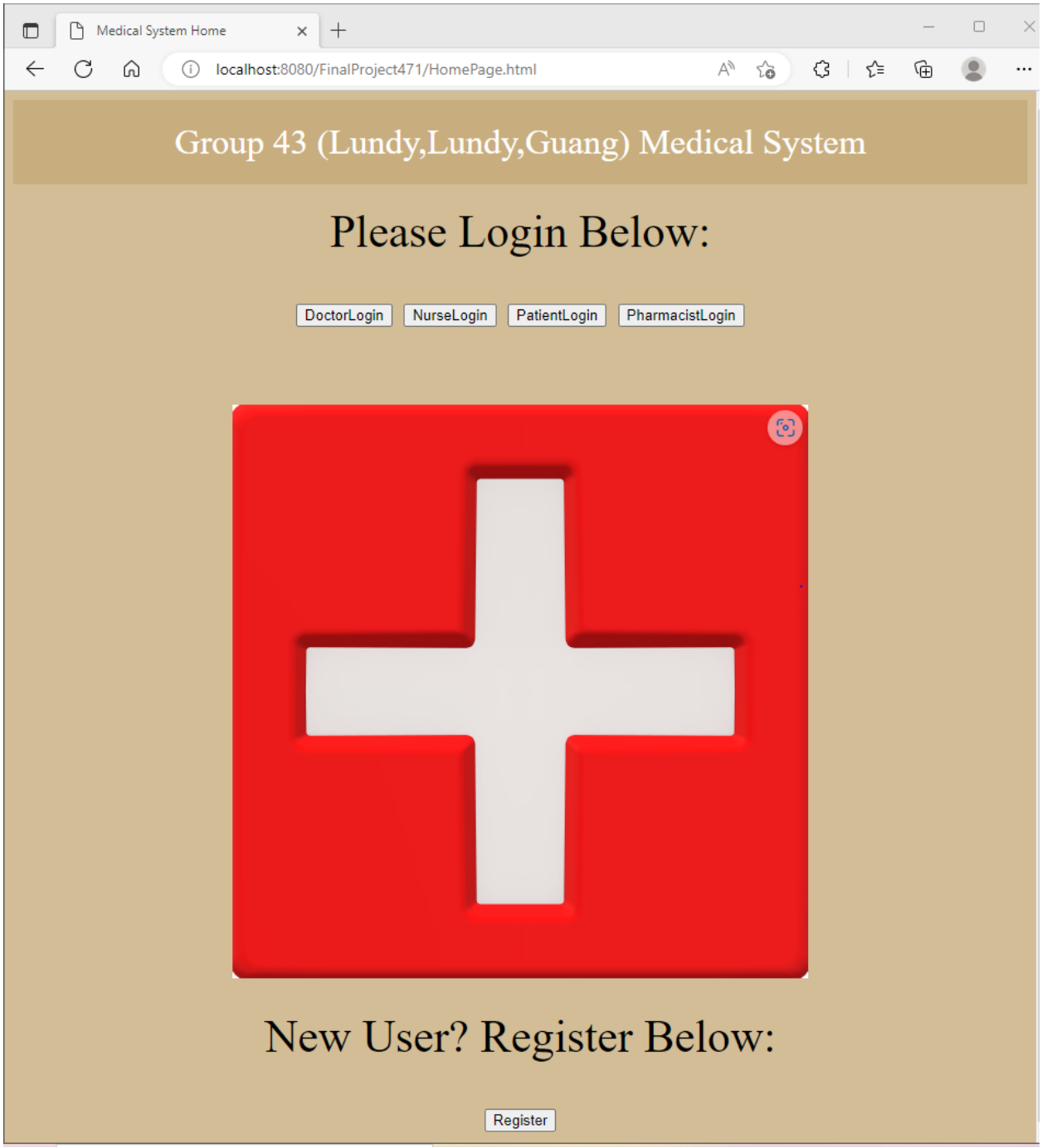


Figure U.1: The homepage of our website (Source code: HomePage.html)

The homepage screen provides a logo, a title for our system, as well as the first two options we will encounter: Login (*above logo*) and Register (*below logo*).
Before starting, refer to *Appendix **U.1*** to view the state of the database *before* this user manual. This will show the sample database population/state created for the user manual, and this state will be manipulated as we progress through this user manual.

We will split this user manual into a part for each user and their functionalities (ie. doctor, nurse, patient, and pharmacist). We will begin with showing the doctor functionality:

1. Doctor Functionality

Let us begin with registering new users. Once users press the register button at the bottom of the webpage, they will be met with this page:

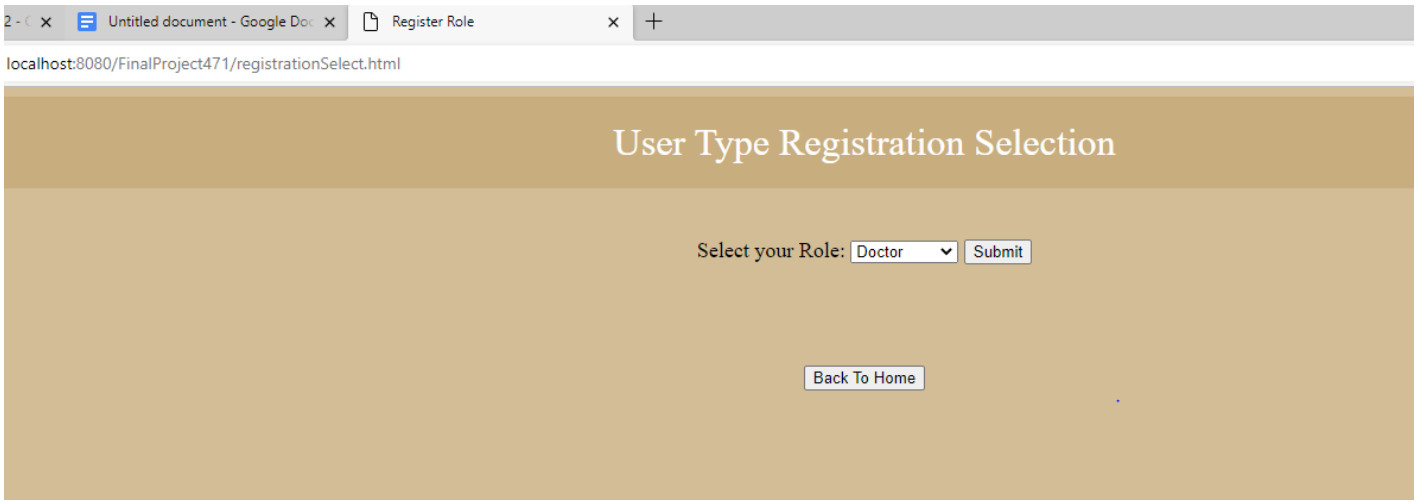


Figure U.2: The user registration page

Here, the user may select their user role using the select role drop down menu. The options are:

Figure U.3: Registration user options

For the discussion of registration, the same functionality persists regardless of role. Let us select doctor first and submit. Once submitted, the next page prompts the potential doctor to fill out their information in a form as shown below:

Figure U.4: Doctor Registration form. Note that Back to home button returns you to the homepage

The first outcome to discuss here is registering a new doctor that does not yet exist in the database, and thus will successfully be added to the database system. Let us fill in the form as such: *ID=215403, FirstName=Johnny, LastName=Appleseed, Username=Lakers248, Password=Calgary403, Phone: (403) 555-9898, Email=JohnApple@gmail.com, none for specialty, and Oasis Clinic as clinic.*

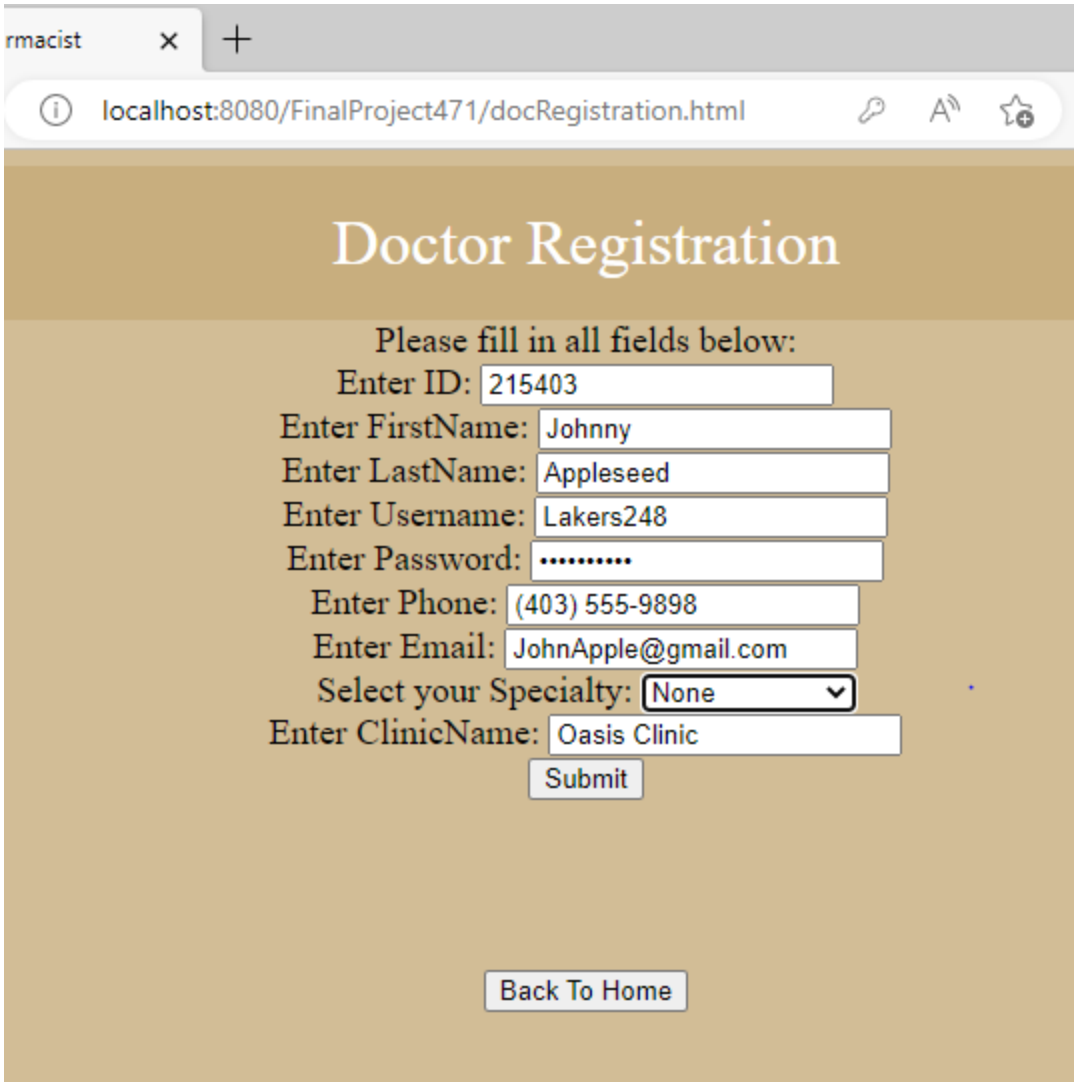


Figure U.5: Form for doctor. Note that Back to home button returns you to the homepage

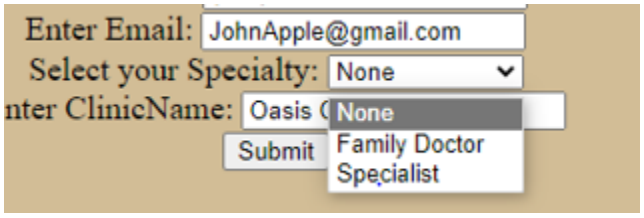


Figure U.6: All options for specialty dropdown

Once Submitted, we expect to see the doctor homepage for that newly registered doctor, shown below:

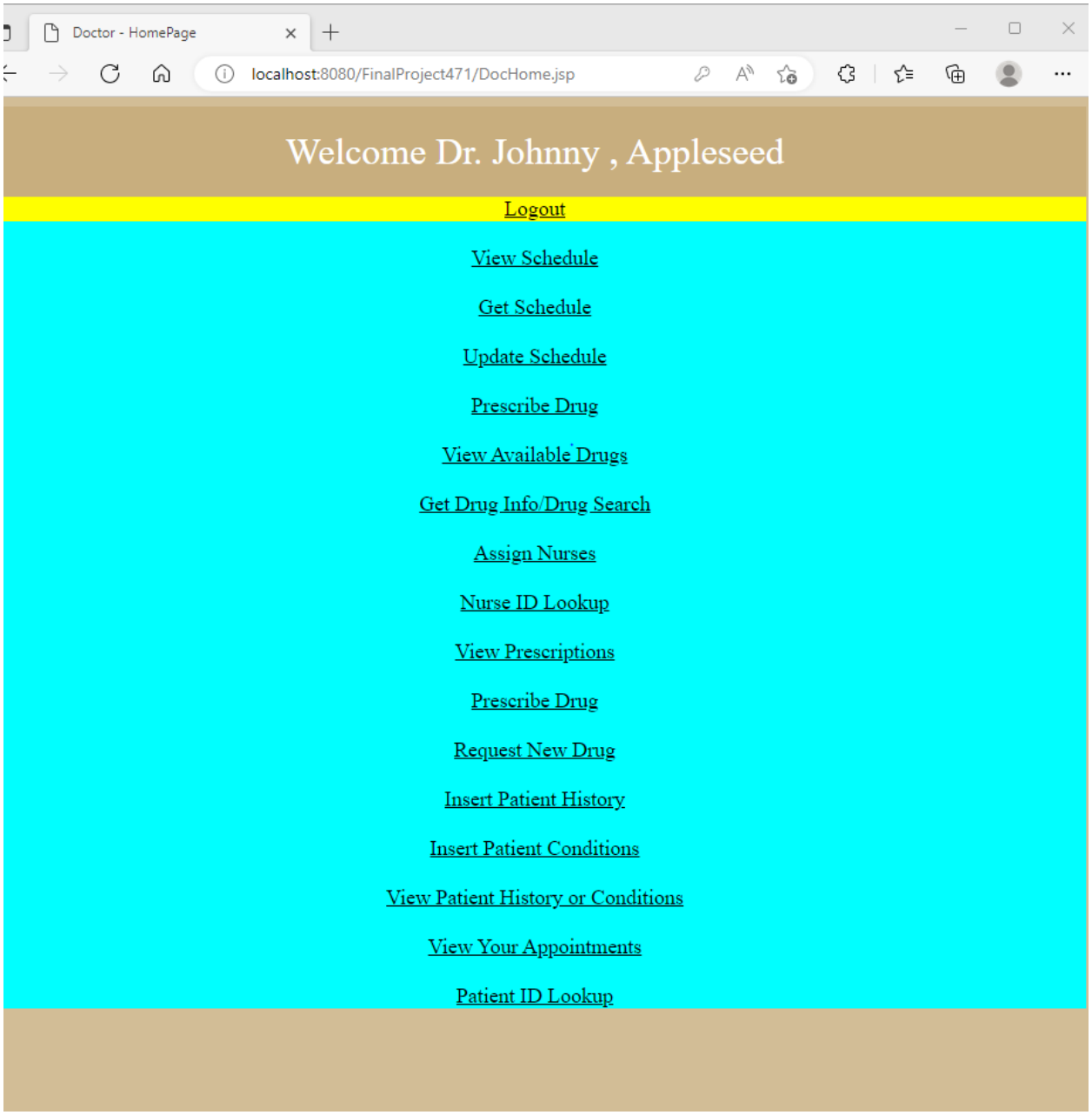


Figure U.7: Doctor Homepage

Here, we see the dynamic webpage has created the user, and shows us all options that this new doctor can do. **Refer to Appendix U.2.1 to view the database change.* The first option to show is the logout, and then we will demonstrate an invalid registration in our system before returning to this page.

Pressing the logout link (top of figure U.7), we return to the homepage, indicating a successful logout.

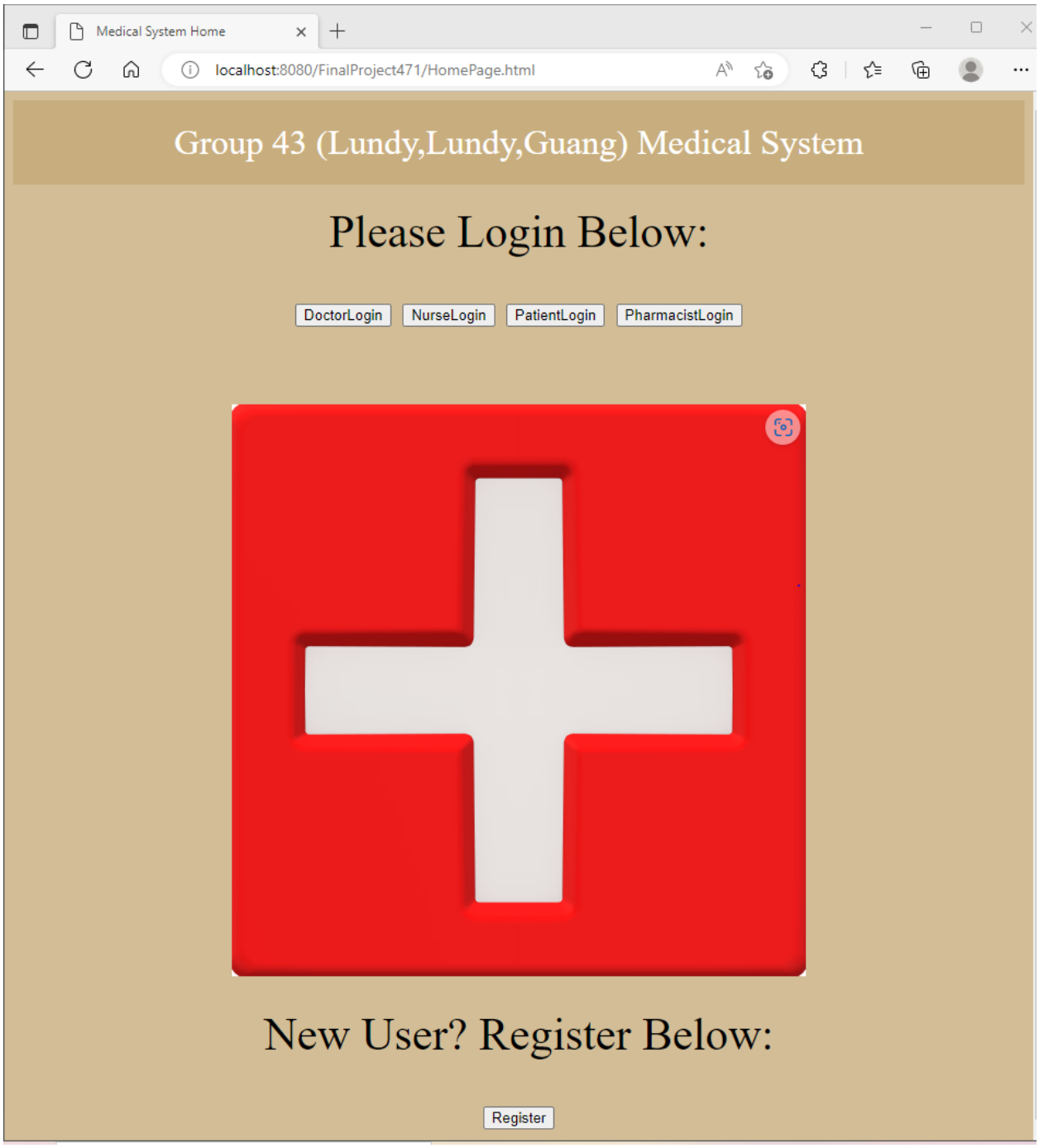


Figure U.8: Homepage after logging out

Now let us register a doctor in an invalid way: attempting to pass NULLs.
To prevent NULLs in our database system, all form fields have been set to required, and therefore must all be filled before submitting. *This is true for any form in our system.* This is demonstrated in *appendix U.3.* Therefore, our system is protected from invalid registrations and

missing information, which is critical to maintain the dependency and integrity of our coded session attributes set at login and registration.

Now we can discuss the login functionality for doctors in our system.

To login, the user must first press the doctorlogin button at the top of the homepage(refer to figure u.8). Pressing the button leads to the doctor login page:

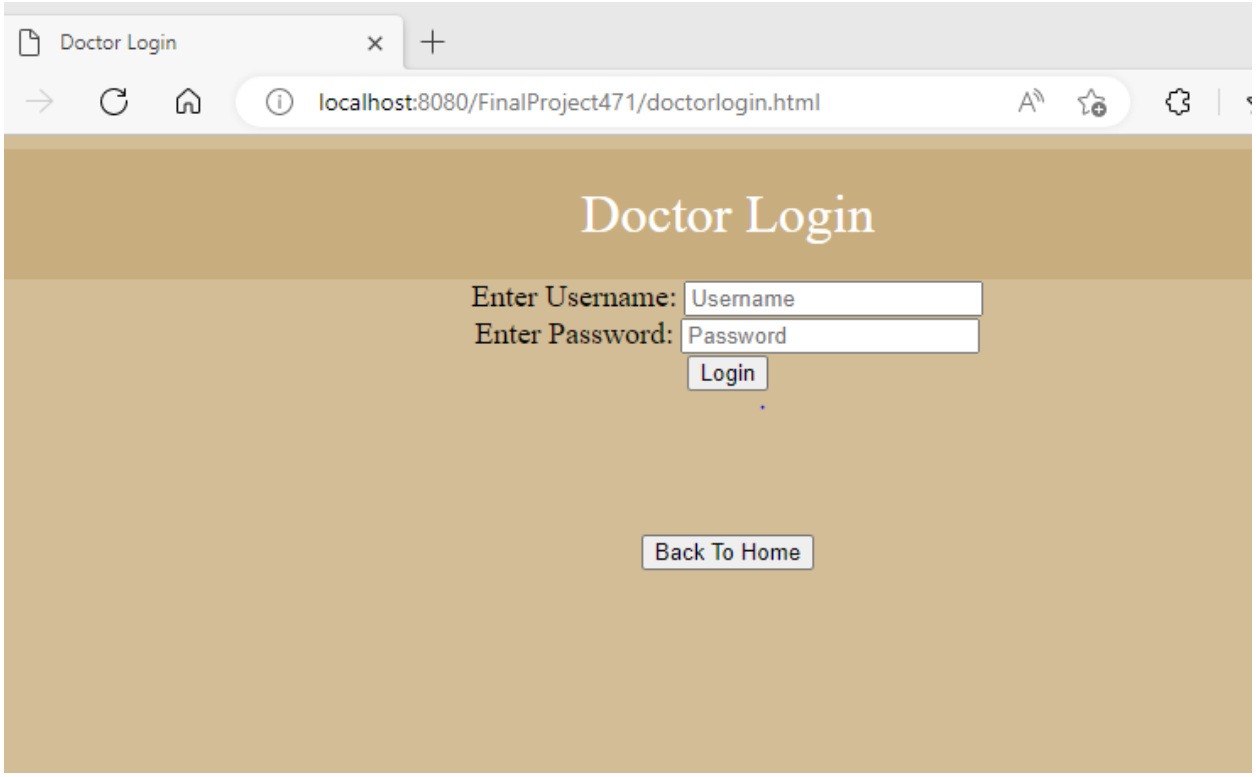


Figure U.9: Doctor Login. Note back to home button brings you back to the homepage seen in figure U.8.

The two options here are an invalid login (logging in with someone who does not exist in the system) shown in figure U.10, and a successful login shown in figure U.11.

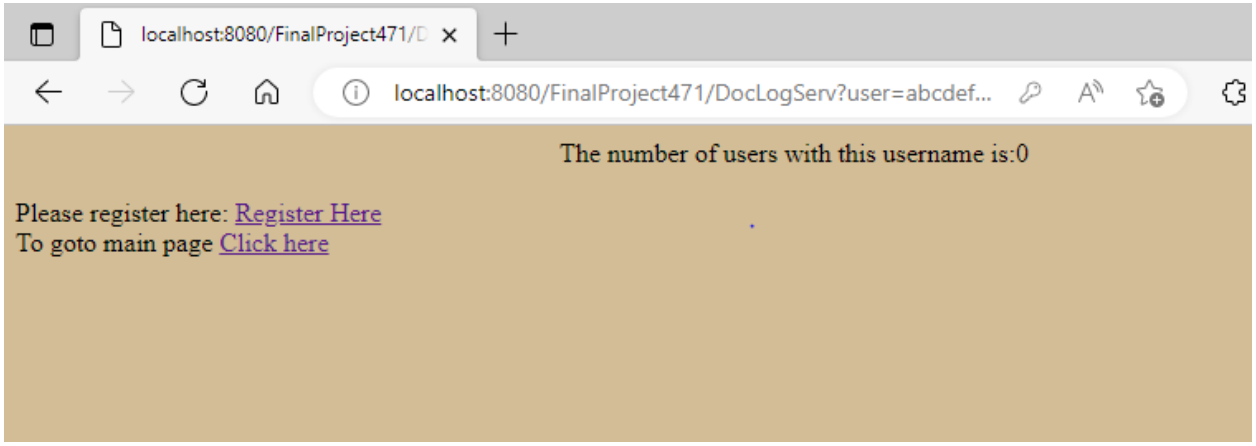
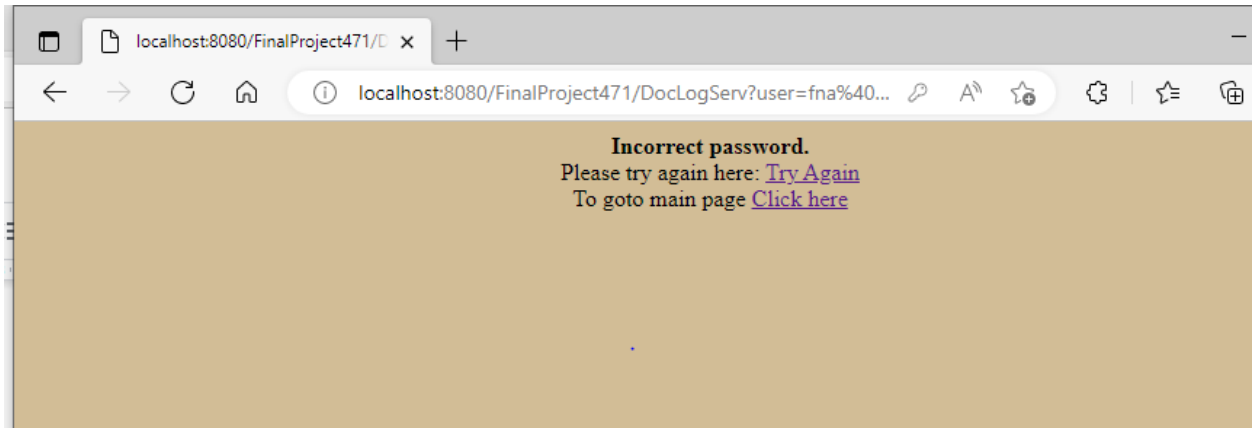


Figure U.10: An invalid login using 'abcdefghijklmnop' and '123456789' as username and password, respectively. Register here link brings you to figure U.5 to register, while Click here brings you back to the homepage of figure U.8.



A valid username, but incorrect password attempt

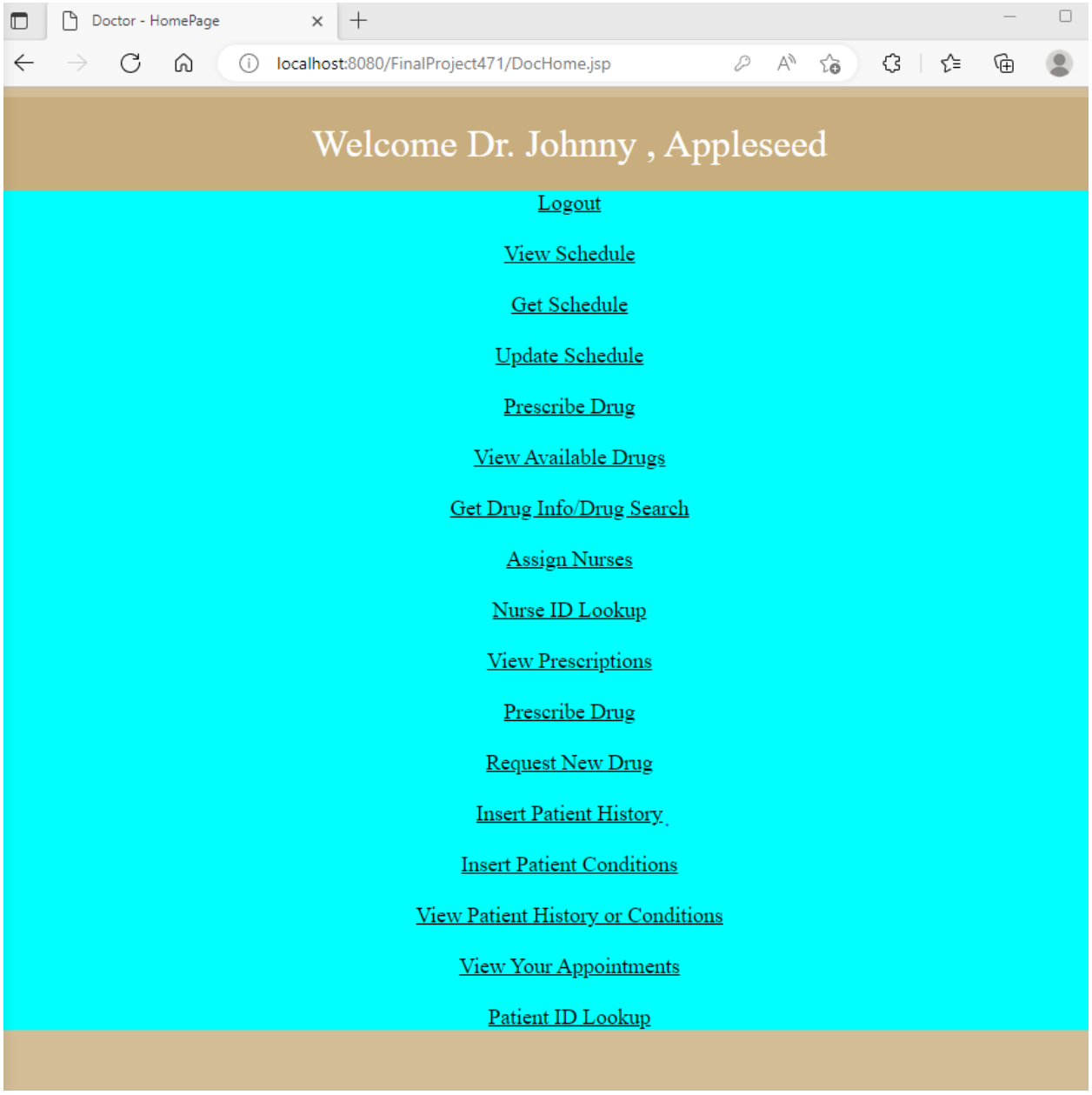


Figure U.11: Logging in with 'Lakers248' and 'Calgary403' username and password. (This matched the user we just created previously, Johnny Appleseed)

Now logged in with Johnny Appleseed, we can demonstrate the existing doctor functionalities/options seen in the previous figure.

***Note: at the time of this user manual, 'View Schedule' and 'Get Schedule' are duplicates, and view schedule does nothing. Therefore, View Schedule will be taken out in the final code submission.

Now we can demonstrate the schedule functionality of doctor, which also corresponds to the doctor_schedule table in our database.First we will look at retrieving a schedule.

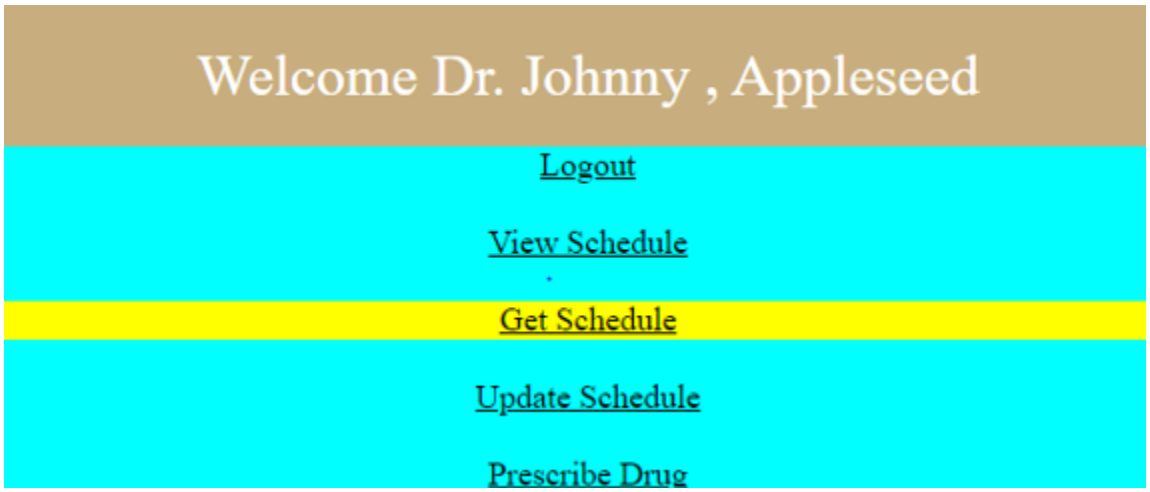


Figure U.12: Choosing Get Schedule

At first, Dr.Appleseed will have no schedule, and thus an empty table will be displayed:

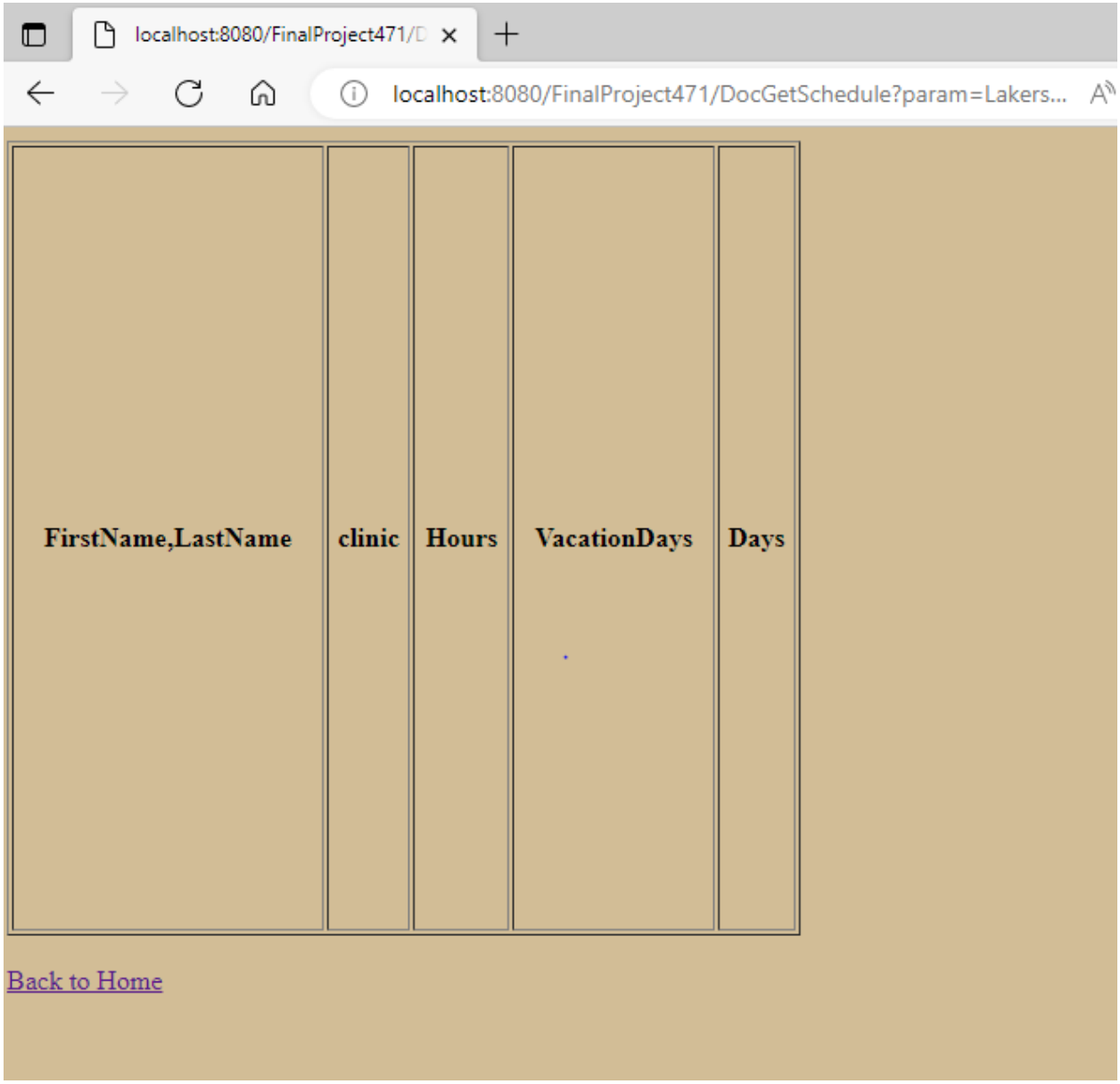


Figure U.13: Empty Doctor Schedule

This, of course, corresponds to the fact that no tuple in doctor_schedule exists yet in the database (**Appendix U.1.7)*.
Now, lets go back (Back to Home Link). Note that this back to home brings you back to the logged in user's homepage.

Let us now select update schedule to make a change to Dr.Appleseed's schedule:

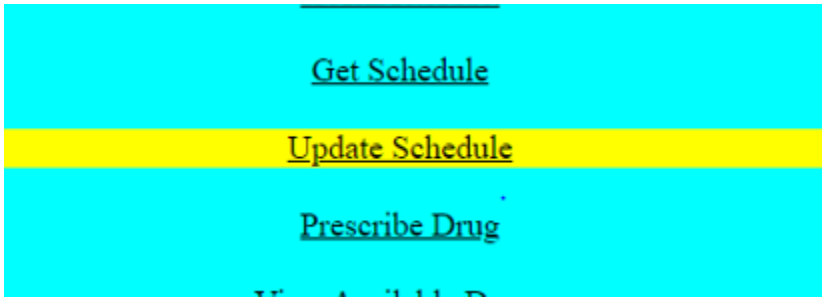


Figure U.14: Selecting update schedule

The update schedule selection prompts this page to appear:

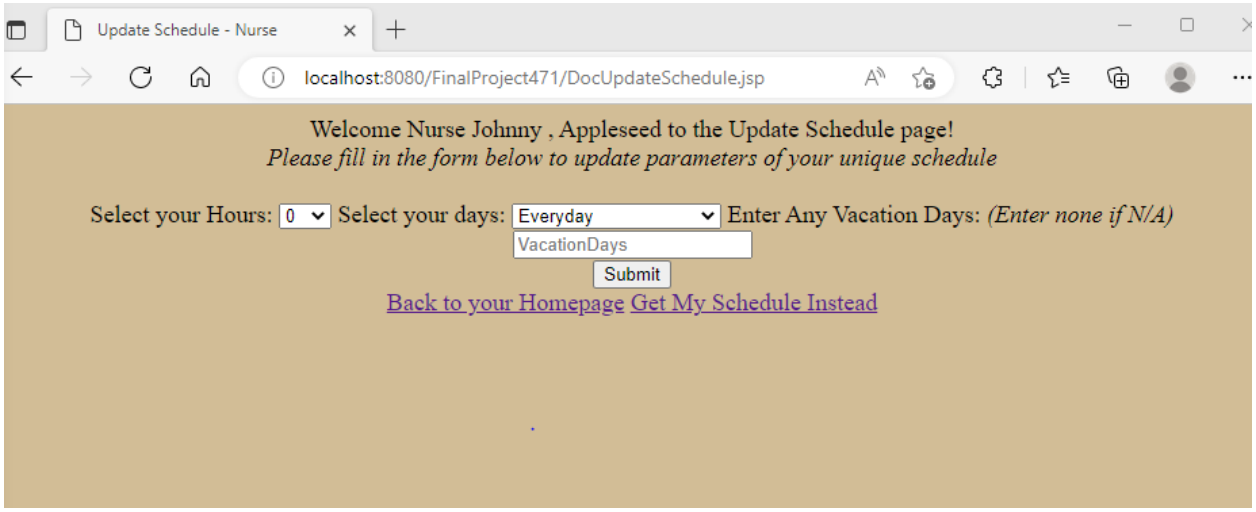


Figure U.15: Update Schedule Form

The form allows for the selection of various hours (figure U.16) and various days (figure U.17) and a text submitted vacation days. The links 'Back to Homepage' brings you back to the user's homepage, while 'get my schedule instead' brings you to the get my schedule page previously discussed.

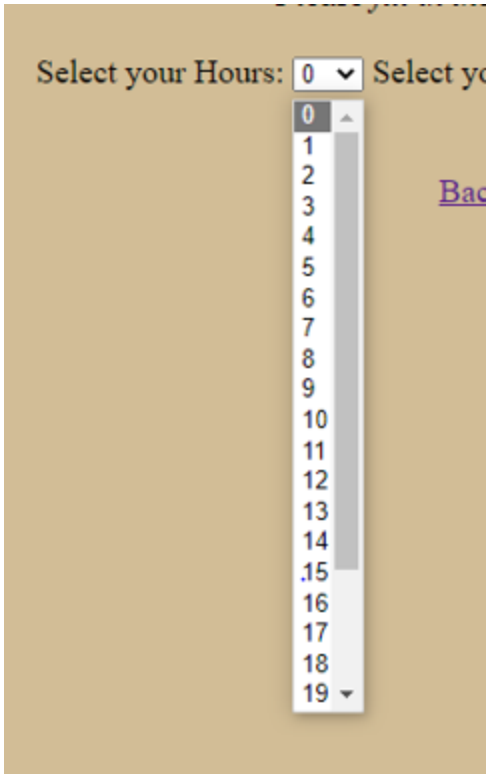


Figure U.16: Dropdown options for hours

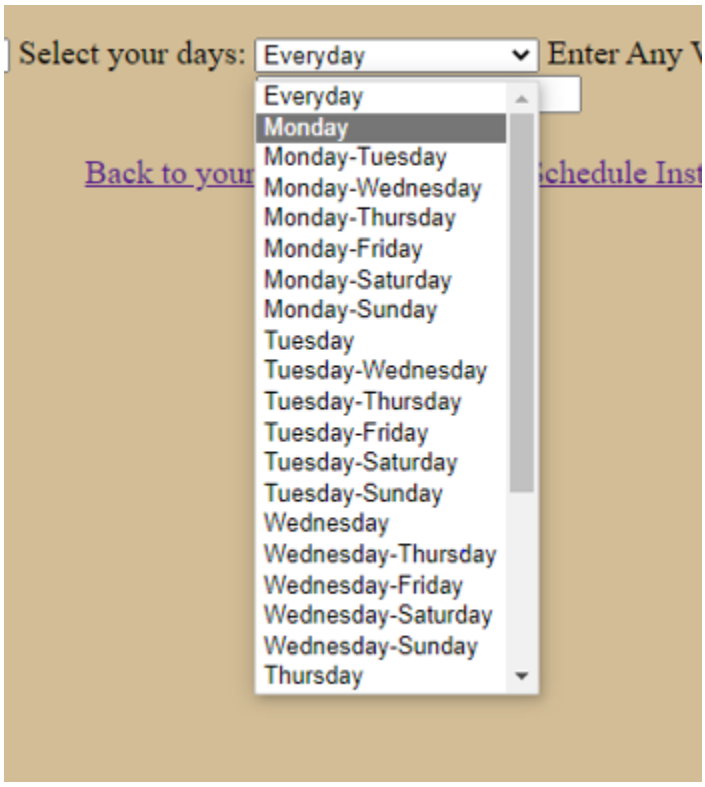


Figure U.17: Dropdown options for days

Let us fill in the form as such:

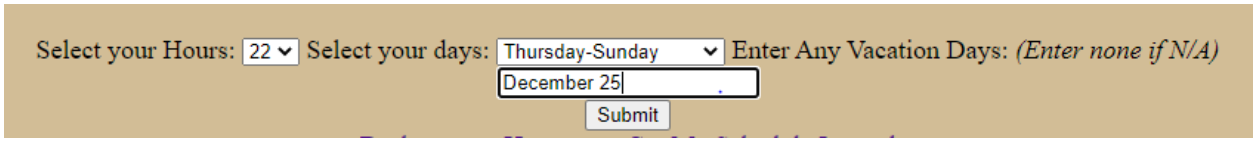


Figure U.18: Sample form for updating doctor schedule

Once submitted we see the success message: (refer to ***_Appendix U.4_ for database change)

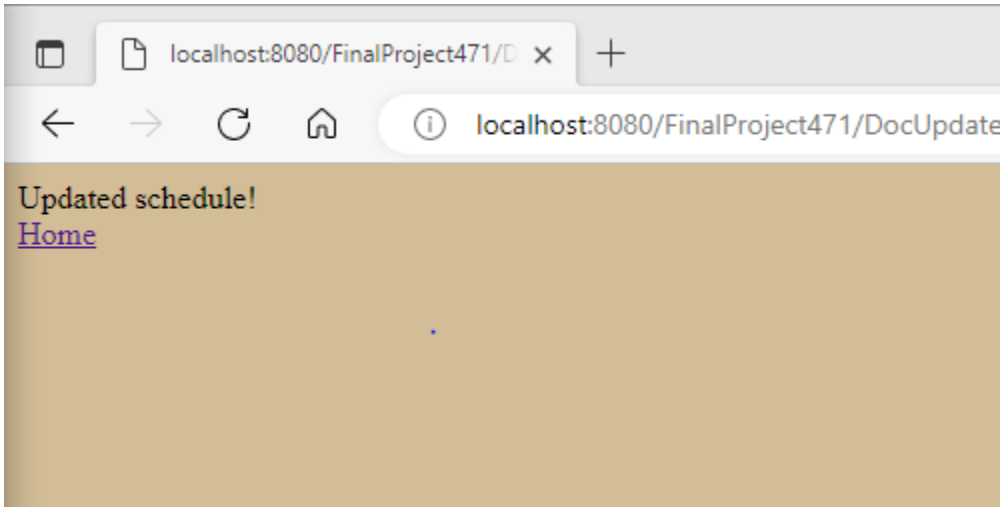


Figure U.19: Successful updated schedule

Now, lets press Home and select update schedule again with these parameters, which will overwrite the previous schedule for this user (***_Appendix U.5)_

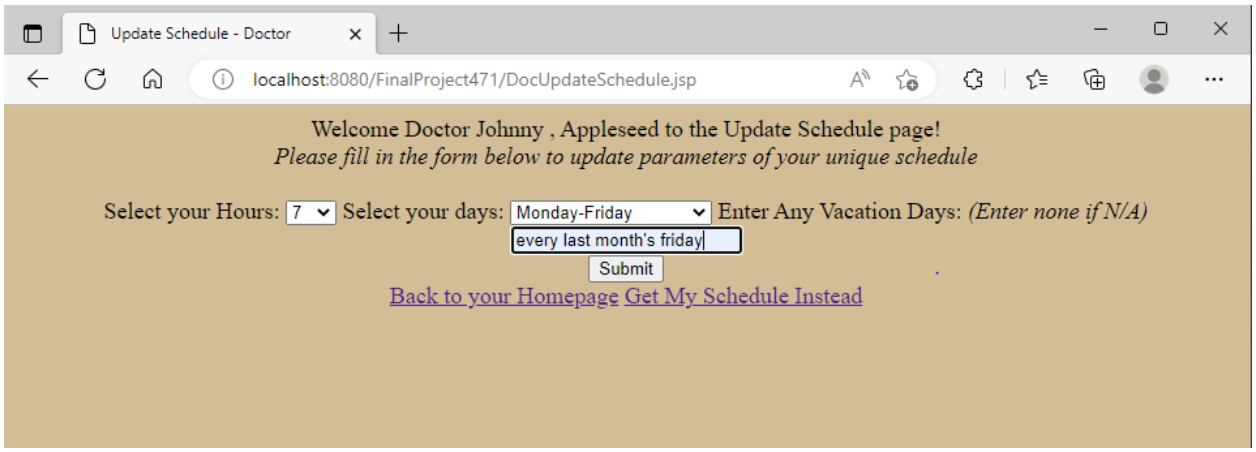


Figure U.20: Updating again to show overwriting previous schedule

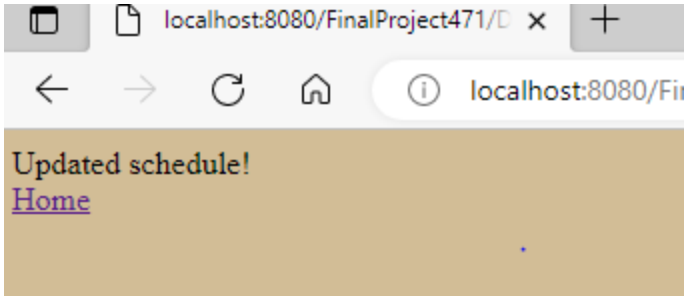


Figure U.21: Successfully overwritten

Now we can show Get Schedule to retrieve an existing schedule. Selecting ‘Get Schedule’ from the homepage (as in figure U.12) now shows this:

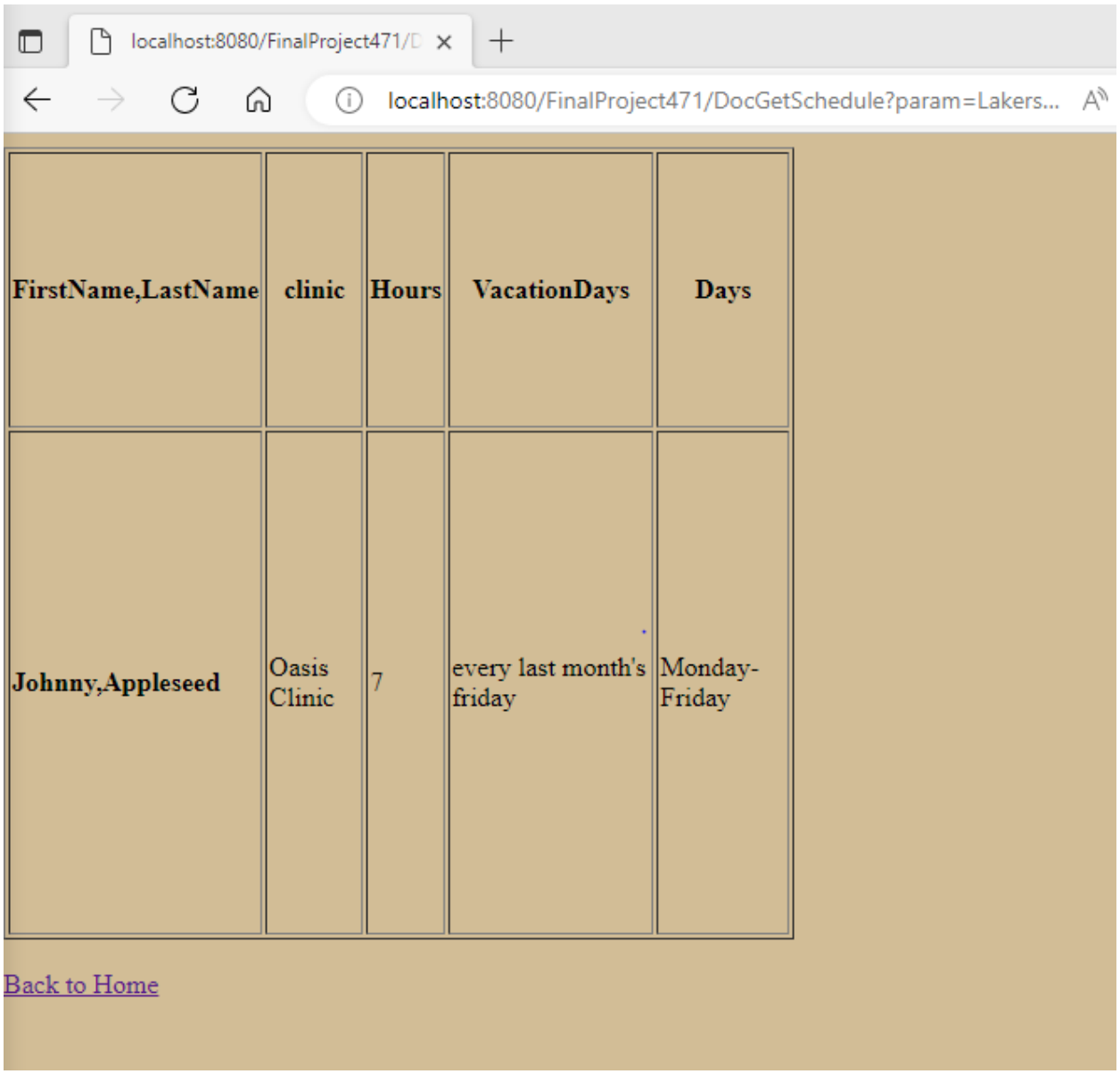


Figure U.22: Retrieved Schedule via Get Schedule

Notice that the most up to date record from update schedule (ie. figure U.20) is shown due to our system's overwriting policy to avoid multiple schedules.

Following the schedule functionalities, we can discuss all the drug functionalities our system provides to a doctor.

This section will include discussion of the options (from figure U.11): View Available Drugs, Get Drug Info/Drug Search, View prescriptions, Prescribe drug, and request new drug.

Logically, we will first explore the view available drugs option to see the drugs available in our system.

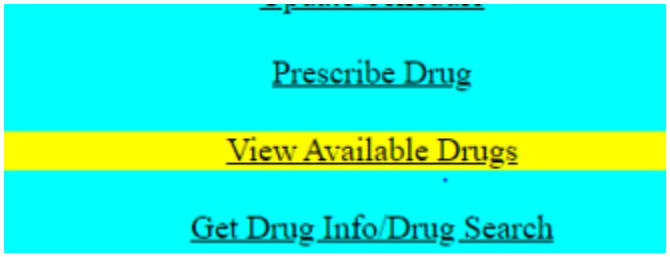


Figure U.23: View Available Drugs option

Pressing the above link yields:

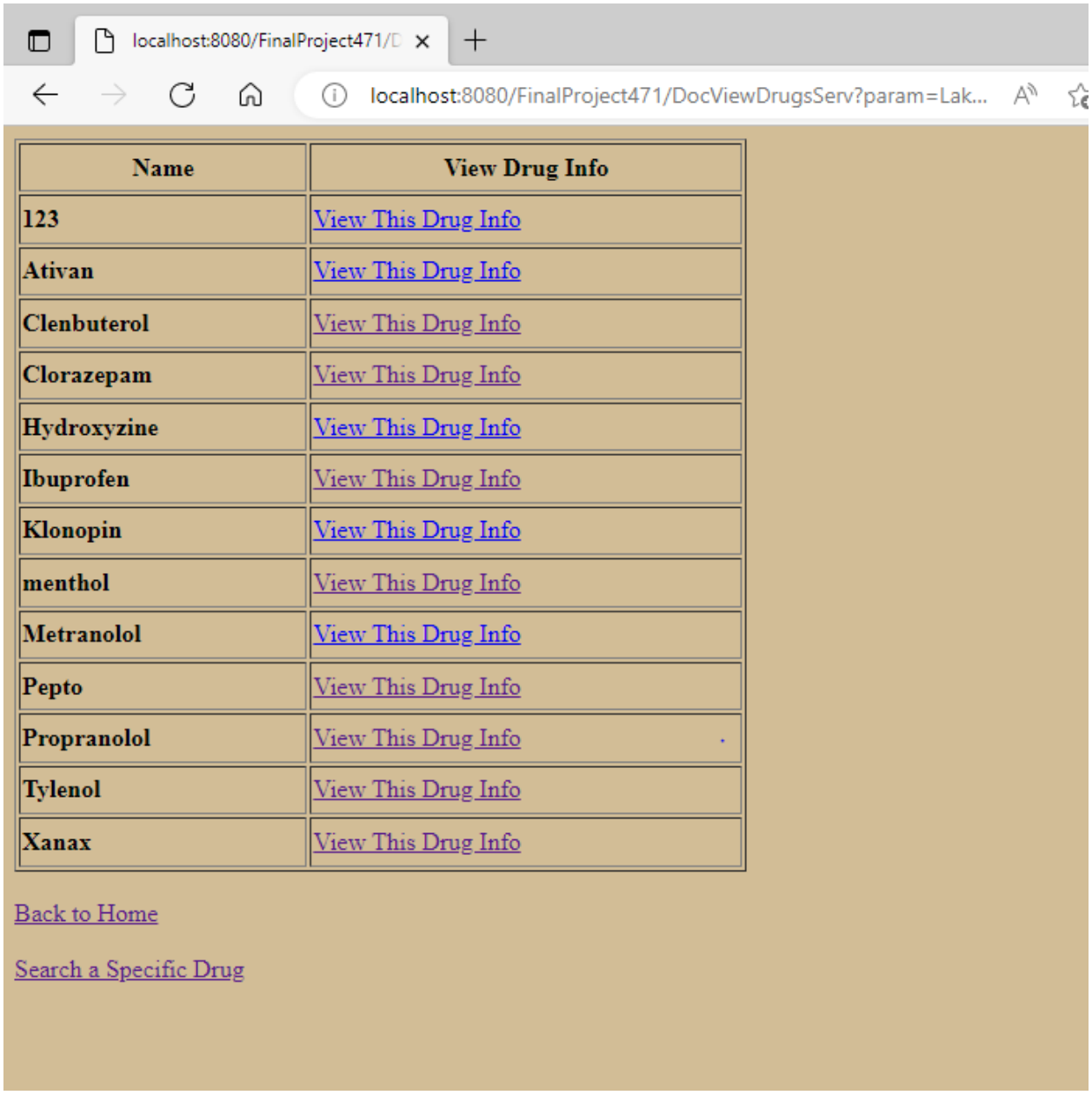


Figure U.24: Available Drugs

Here, we see all available drugs registered in our system (refer to ****Appendix U.1.8*), as well as various links. For every drug name in the table printed, there is a link to view this drug info. Pressing it shows more detailed information for that specific drug. For example, we can press the link for 'Ibuprofen' as shown below:

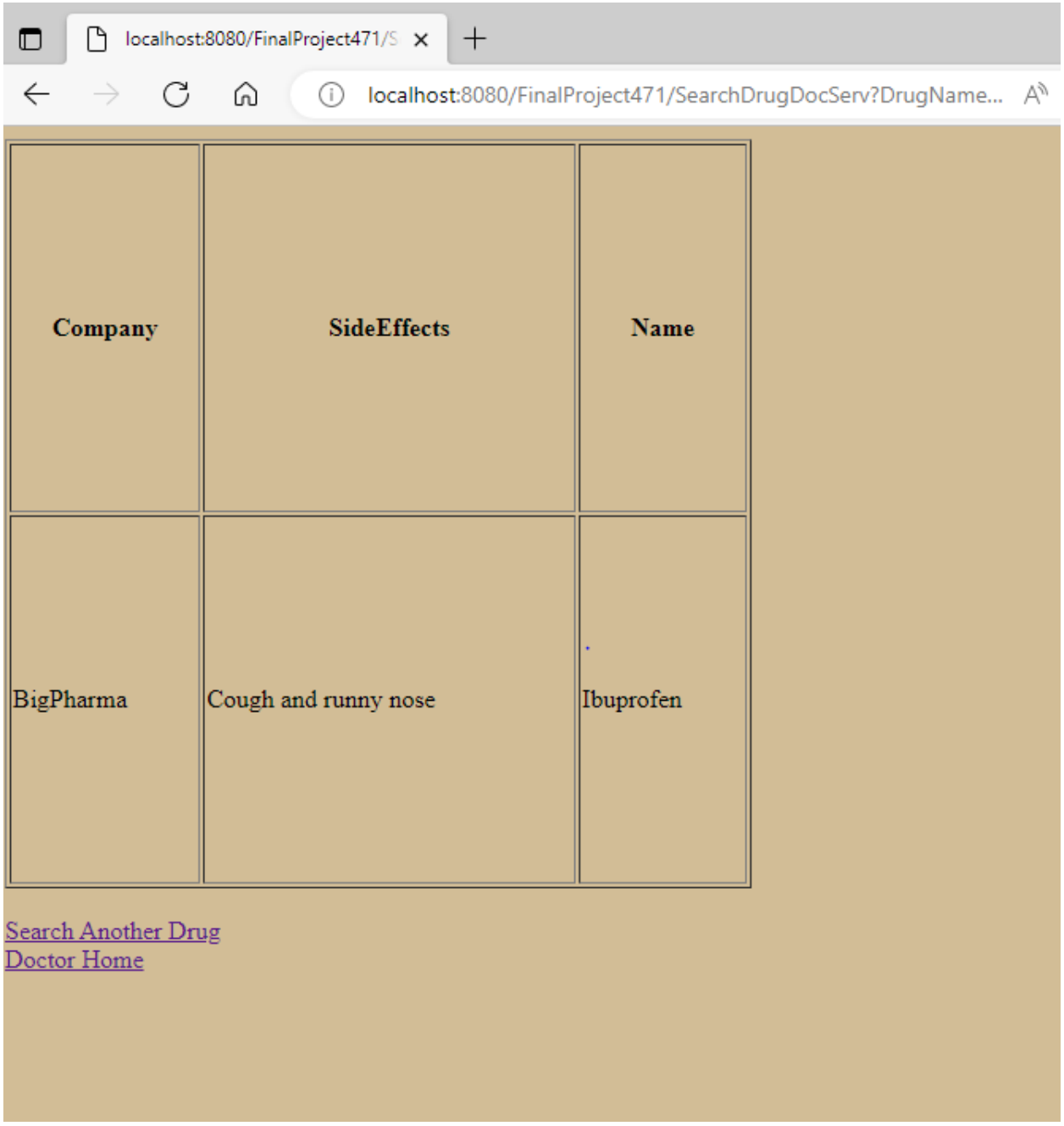


Figure U.25: Selecting 'View This Drug Info' for name 'Ibuprofen'

Viewing drug info shows more detailed information, namely Company and SideEffects, for that specific drug. The link 'Search Another Drug' brings you to the drug search shown in figure U.26 (which we will discuss next), and the doctor home brings you back to the familiar doctor homepage.

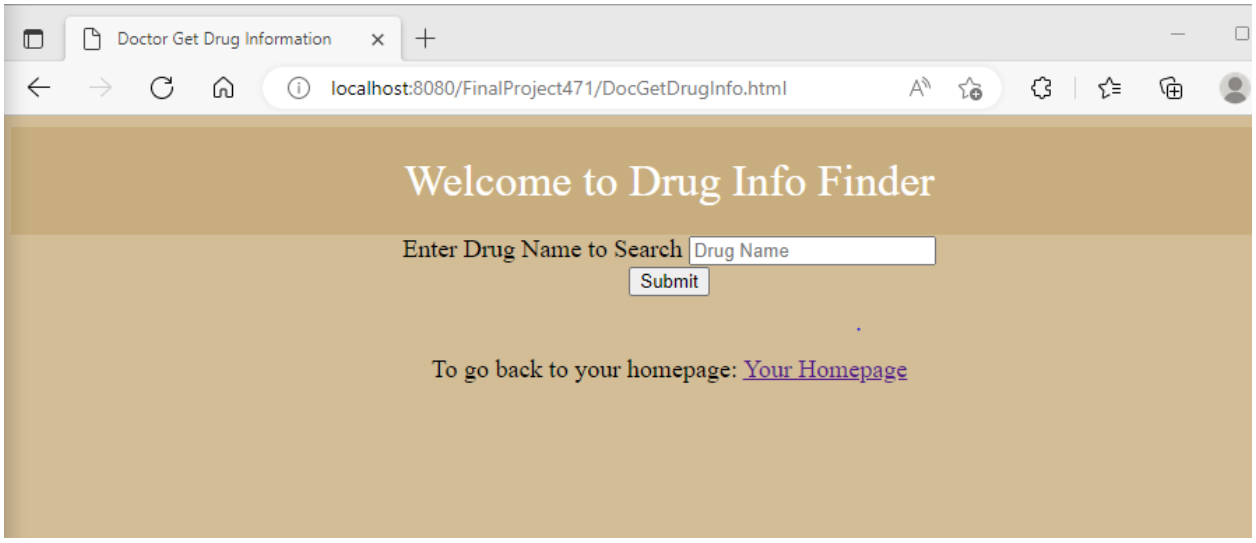


Figure U.26: Drug Search (note: homepage link brings you back to the doctor homepage)

Next, the drug search option seen in figure U.26 is to be discussed. There are 3 pathways to get to this page: (1) Selecting Get Drug Info/Drug Search from the doctor homepage (figure U.27), (2) From the Search a Specific Drug link on the view available drugs page (figure U.24), and (3) Pressing Search Another Drug once viewing a specific drug's info (figure U.25). Any of these options bring you to the above figure U.26.

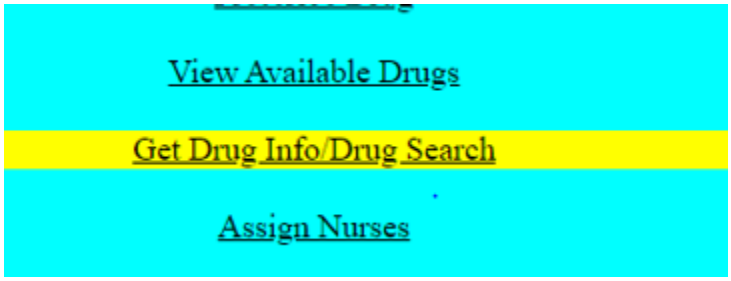


Figure U.27: Drug Search via Doctor Homepage

The purpose of the Drug Search is to be able to view the specific information of a drug by searching its name, and thus provides an alternative to finding the drug in the 'View Available Drugs' table mentioned before.
Searching an existing drug, eg. menthol (figure U.28), yields Figure U.29.

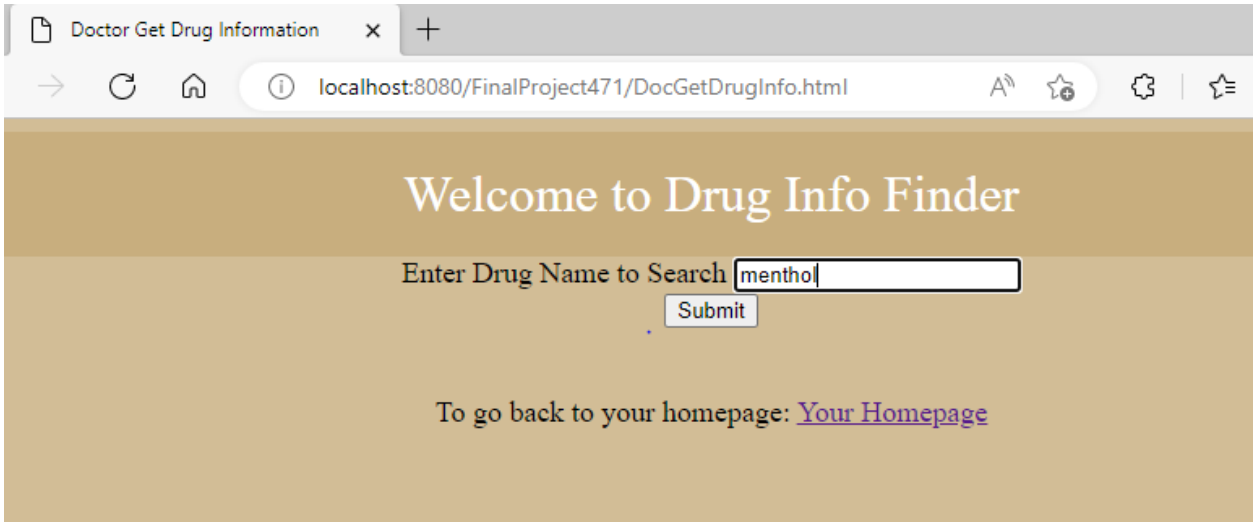


Figure U.28: Searching for existing drug 'menthol'

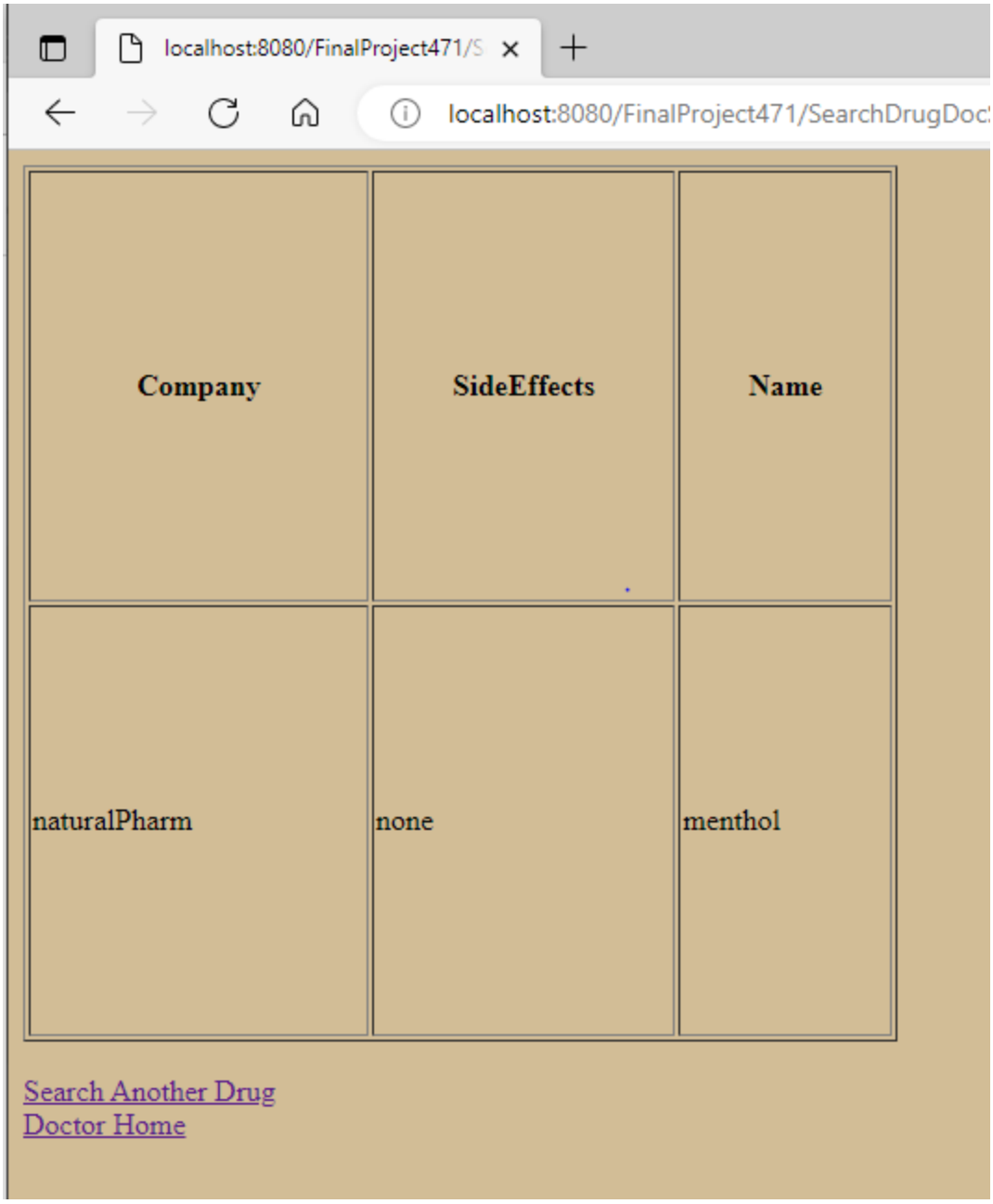


Figure U.29: Search result for 'menthol'

Now lets press the Search Another Drug link to search for another drug, this time one that does not exist.

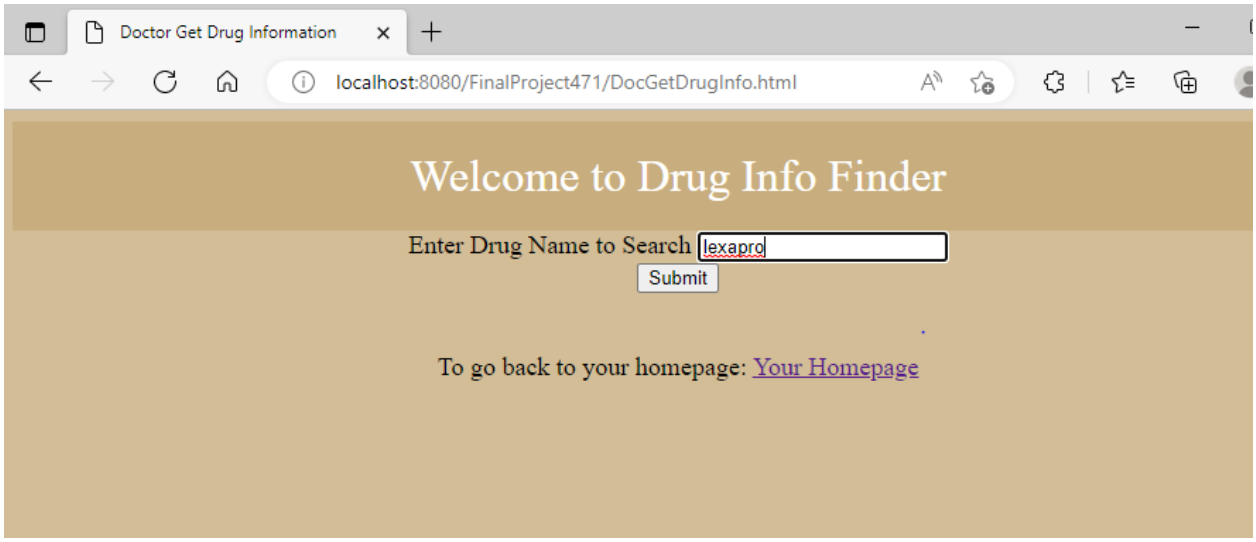


Figure U.30: Searching a drug that does not exist in our database: 'Lexapro'

Which results in an empty table:

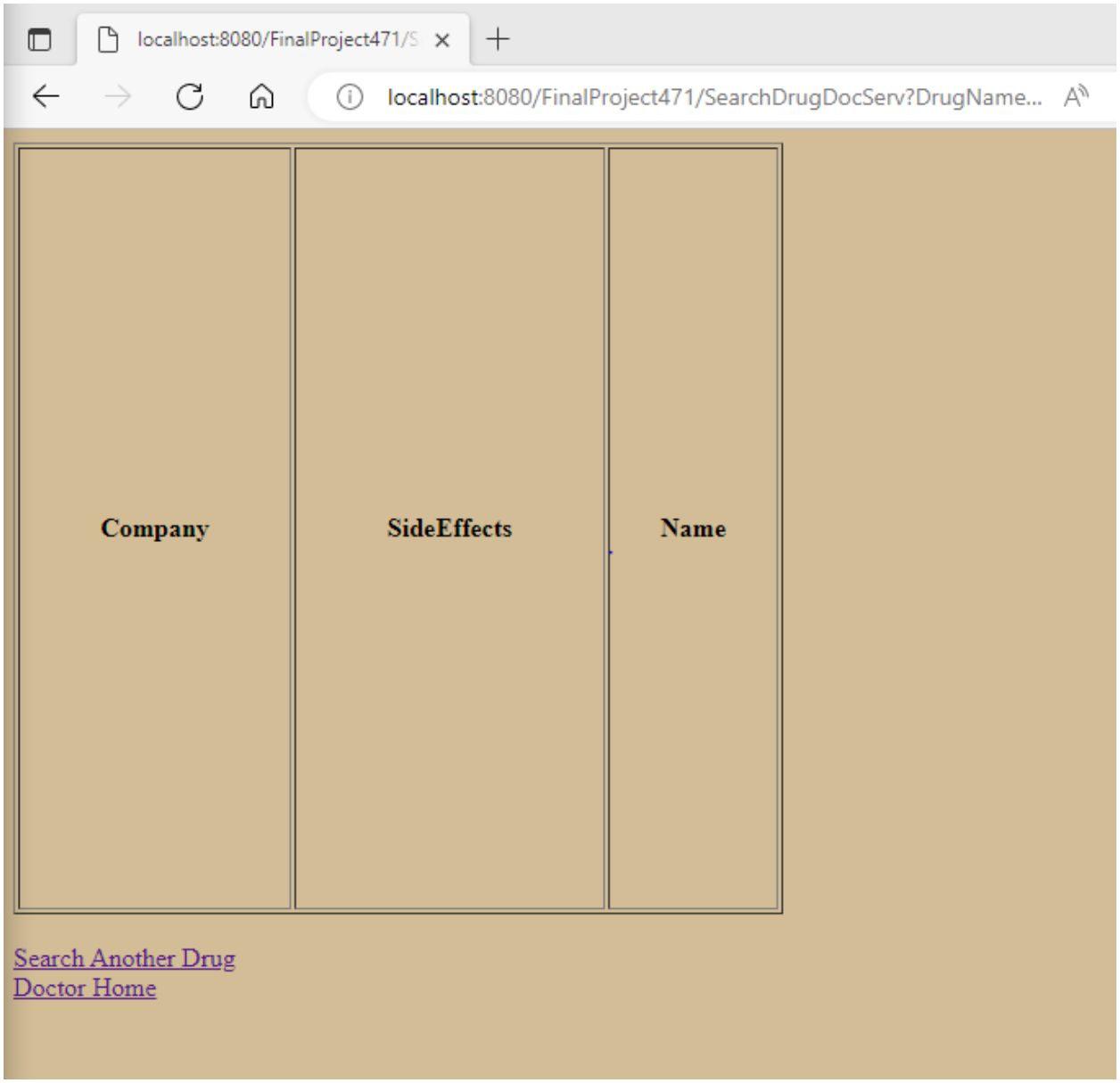


Figure U.31: Empty search for non-existent drug Lexapro (note the use of the term non-existent refers to drugs that are not registered by pharmacists in our database system at that time, and does not indicate whether that drug exists beyond the scope of our system or not)

Now we can discuss the view prescriptions, prescribe drugs and request new drug options. First, viewing prescriptions allows for doctors to view all prescriptions that they have made to specific patients. First, we must select the 'View Prescriptions' option from the homepage:

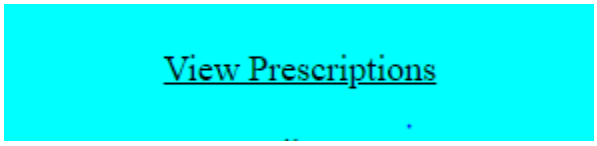


Figure U.32: View Prescriptions option

Which brings us to this page:

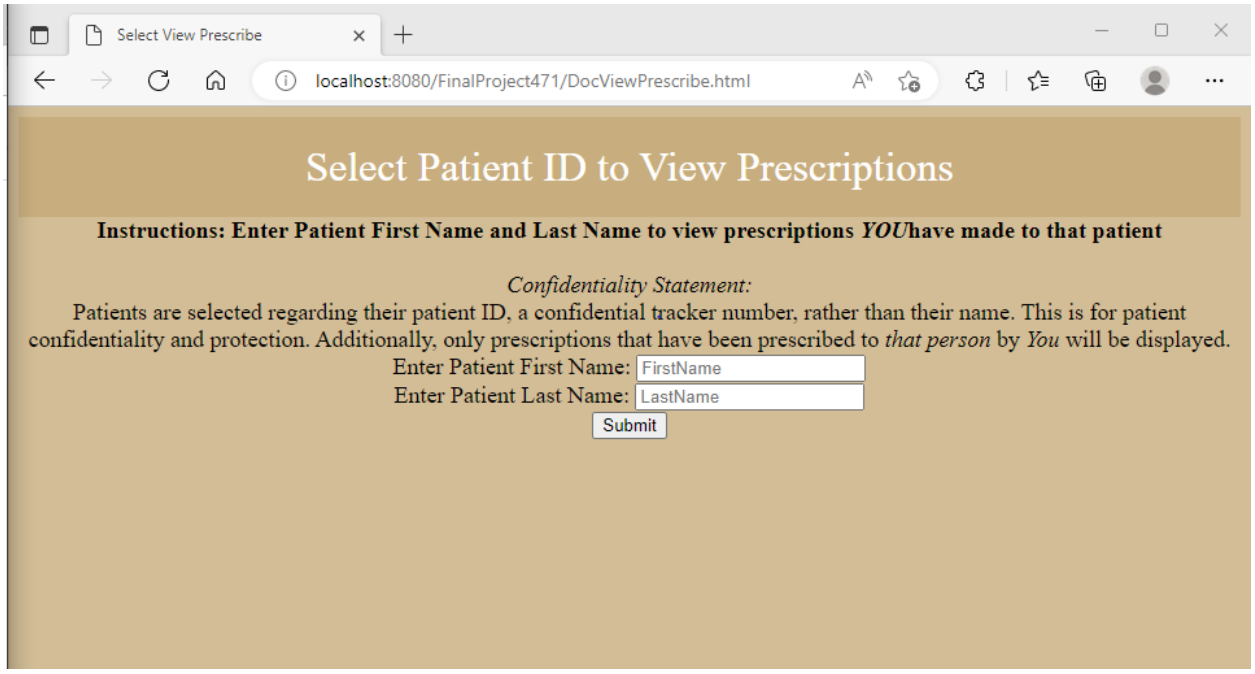


Figure U.33: First Step in viewing prescriptions

Here, the system requires the doctor to enter the first and last names of the patient they are requesting to view prescriptions they made to. For security reasons, the system will only show prescriptions that the currently logged-in doctor made to that specifically searched person. Since Dr. Appleseed has yet to make a prescription, we can show how the system responds to empty or invalid input here. Both an invalid request (entering someone who does not exist) or an empty request (entering a correct patient who has no prescriptions by the logged in doctor) yield similar results:

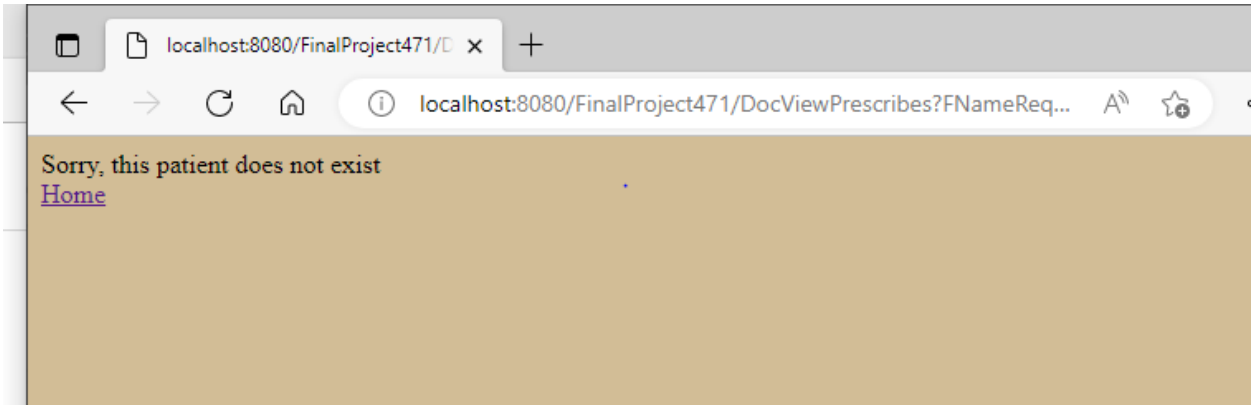


Figure U.34: Entering 'Michael' 'Jordan' as First Name and last name. Since this patient does not exist (refer to appendix u.1.11), we get this message.

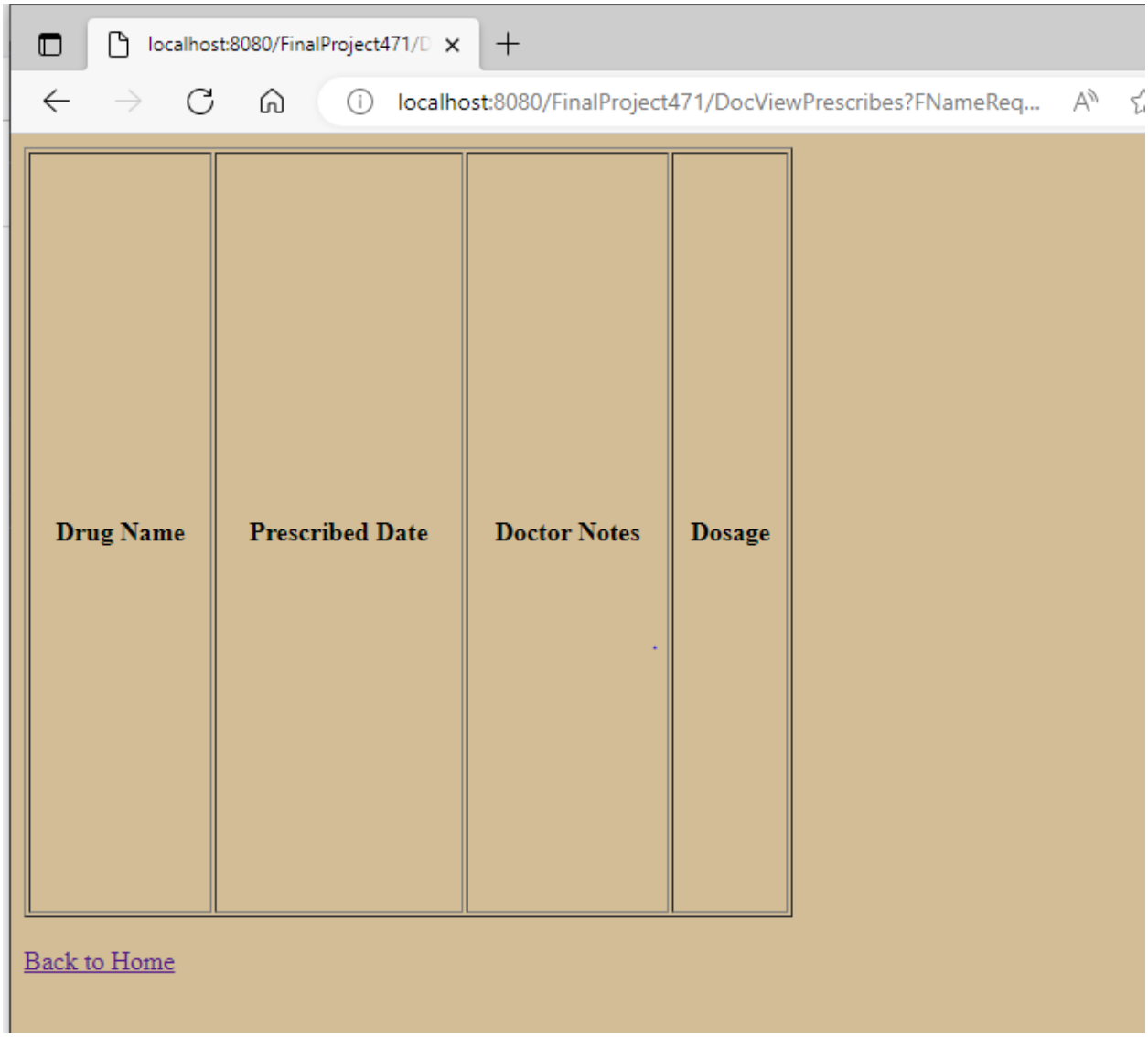


*Figure U.35: Entering 'Frank' 'Ocean' as the patient name. This patient does exist, but does not have any prescriptions from this doctor (***Appendix U.1.14), and thus an empty set is shown.*

In order to view prescriptions without showing an empty set, we must go and prescribe a drug to a patient to see it. To do this, lets press 'Back to Home' and select the 'Prescribe Drug' option:

Figure U.36: Prescribe Drug from Doctor Homepage (NOTE: first prescribe drug option is a duplicate that will be removed at the time of final code submission)

Selecting prescribe drug yields the following page:

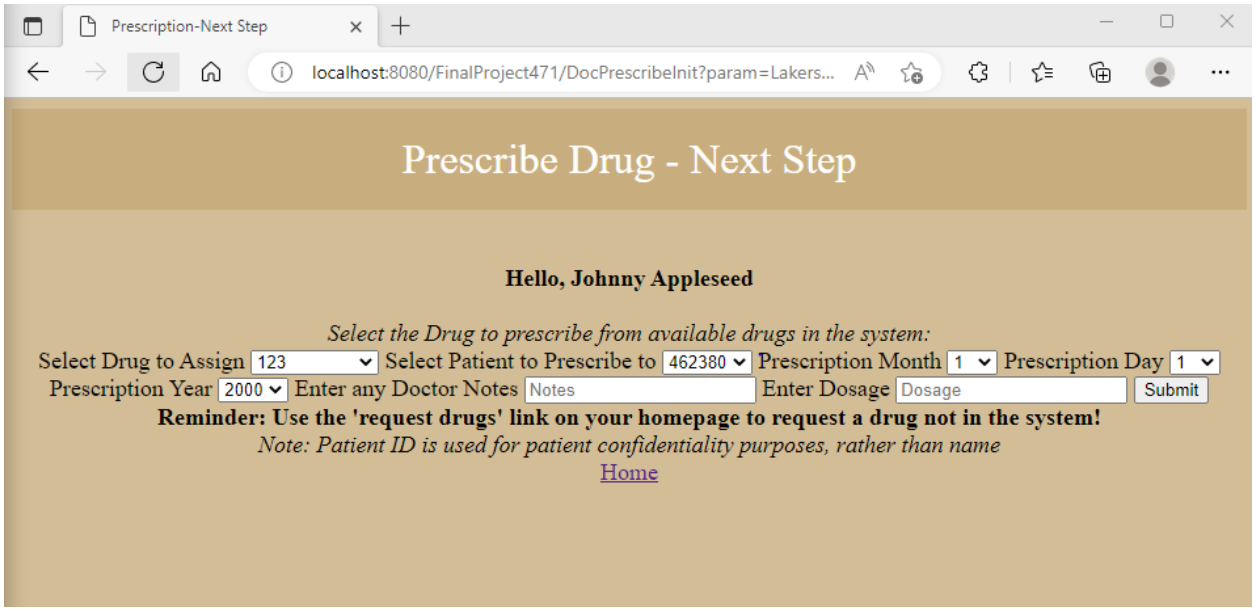


Figure U.37: Prescribe Drug

The resulting form includes a dropdown for the drug name to assign which includes every available drug as an option, a dropdown to select the patient ID to prescribe to (with every patient as an option), the month (1-12), day (1-31), the year (2000-2022), and doctor notes and dosage text fields. Lets fill the form with these example values:

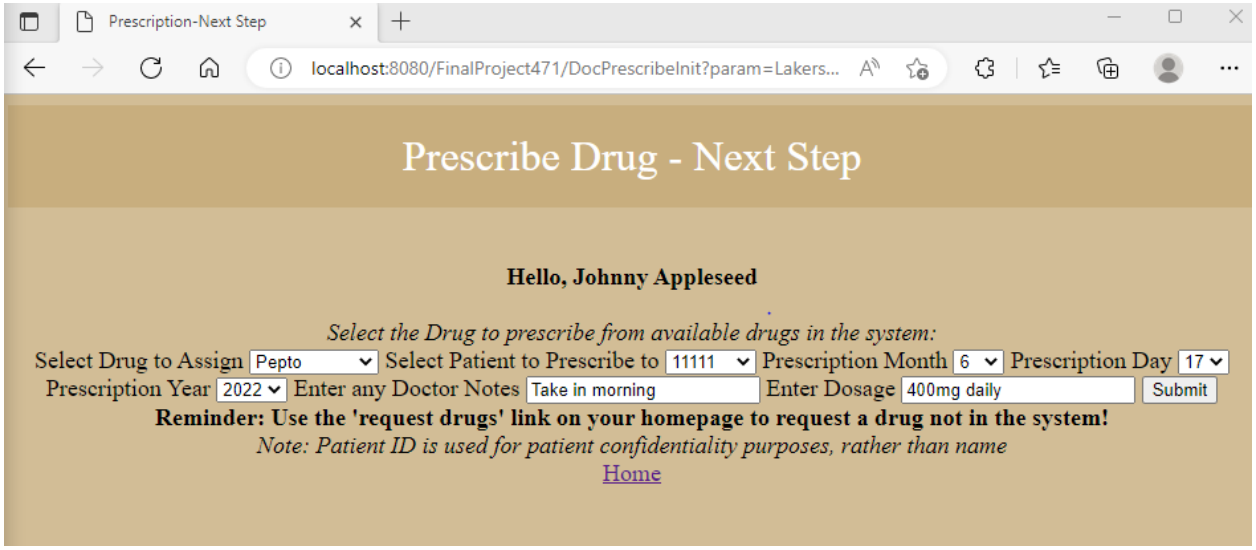


Figure U.38: Sample prescription form entry. Here, we are prescribing to the patient 'Pat Zero' with ID=11111 (refer to appendix U.1 to see this patient in database)

Submitting this form (and thus prescribing to this patient):

Figure U.39: A successful prescription!

The tangible database alterations may be viewed in Appendix U.6. Recall previously that the use of required fields to avoid NULLs applies to every form, including this one. Paired with the dropdown menus for certain fields, this prescription process is protected from unsuccessful prescription occurrences.

Now that we prescribed to patient id=11111, we can go back and view the prescriptions. But first, we can briefly show the patient ID lookup option, which allows doctors to match patient names with their IDs for forms that require IDs (like prescribing).

Figure U.40: Patient ID Lookup option

Selecting this option (from the doctor homepage) yields a simple page as shown below:

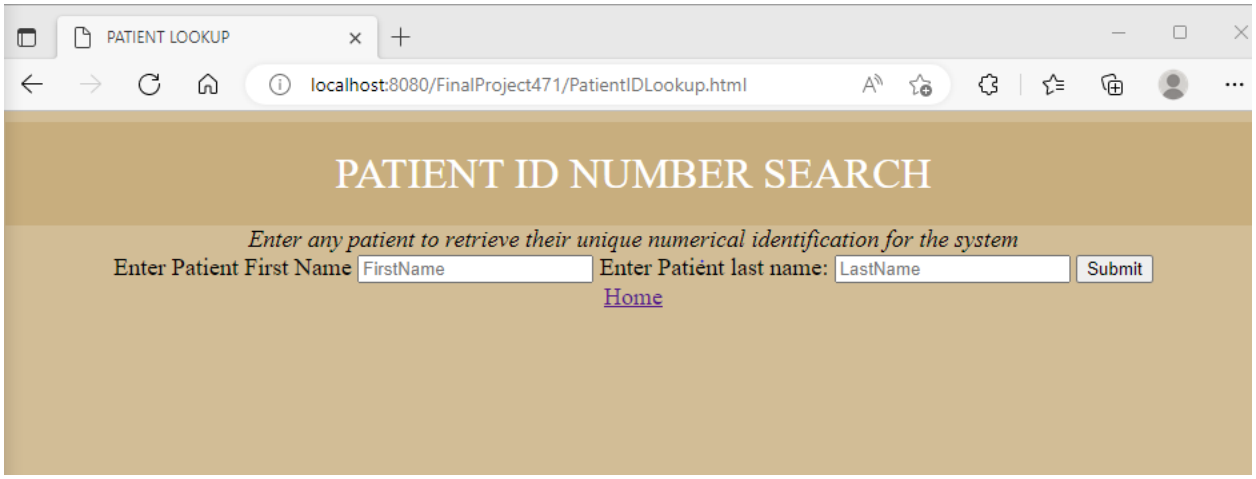


Figure U.41: Patient ID Lookup Search form

This allows us to search for patients' IDs, which is critical for many operations such as prescribing. Lets search up an existing user: Pat Zero, who we previously prescribed to.

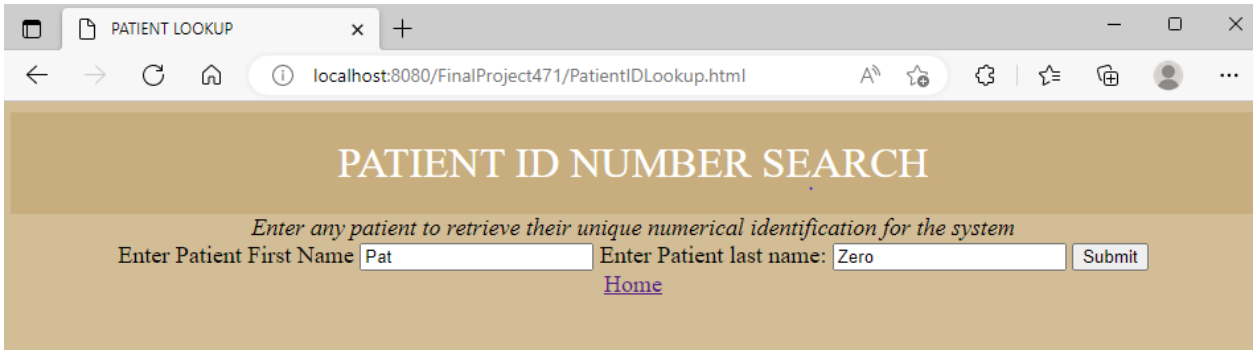


Figure U.42: Searching for Patient 'Pat Zero'

The result of submit is:

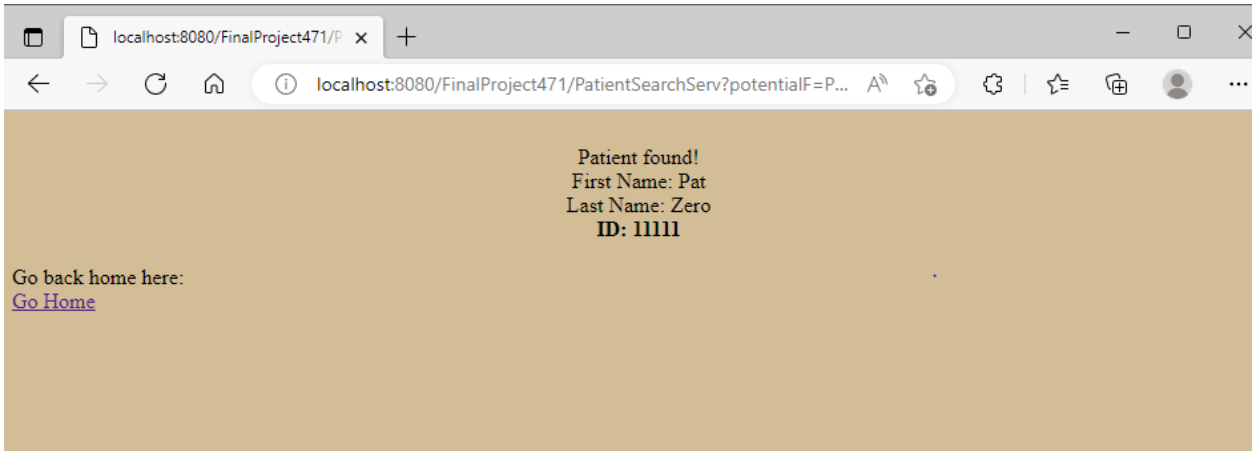


Figure U.43: A successful search!

As seen in the figure above, the search retrieves the ID for that patient that is searched. Going back and searching for a non-existent patient "QWER TYU", the search gives us this:

Figure U.44: An unsuccessful search

Now we can go back and prescribe again to ID 11111 corresponding to Pat Zero, and show the view prescriptions with multiple prescriptions to a patient.

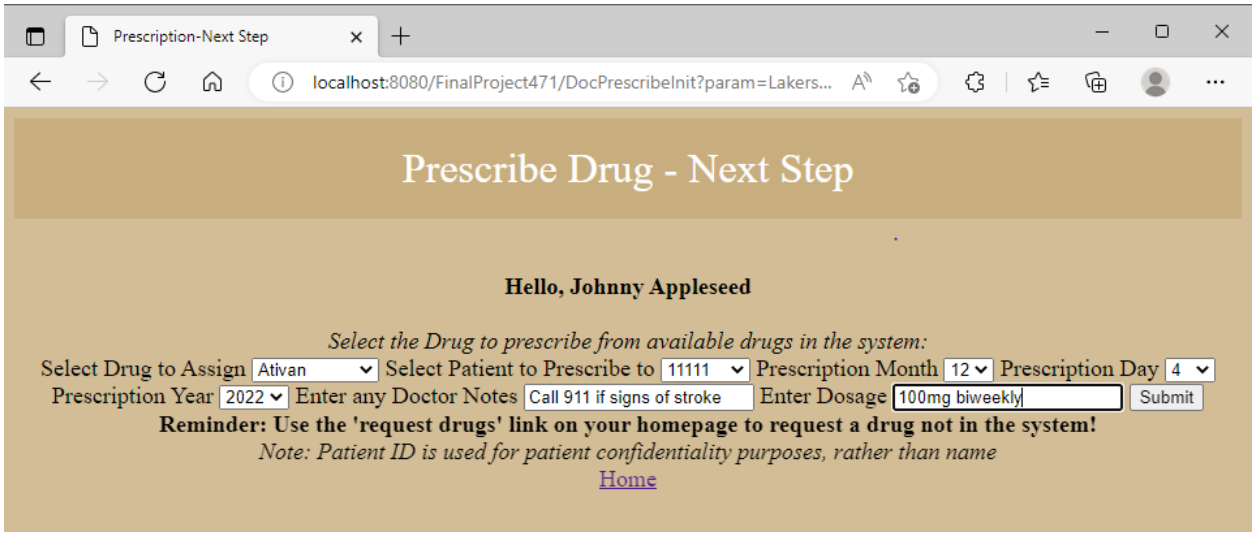


Figure U.45: Another prescription to the same patient

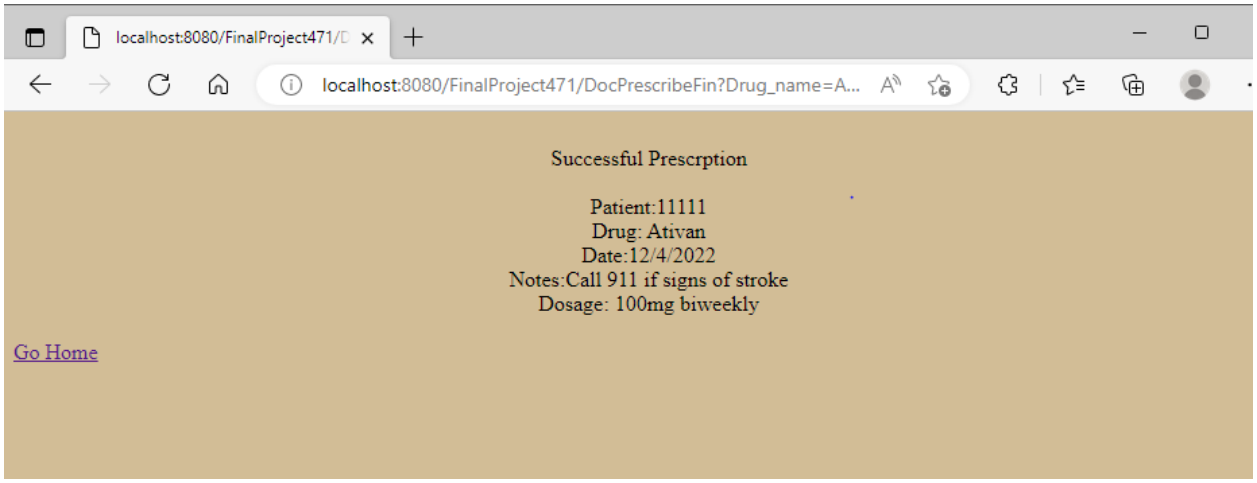


Figure U.46: After submitting figure U.45

Figure U.47: Viewing our prescriptions to Pat Zero now after previous transactions

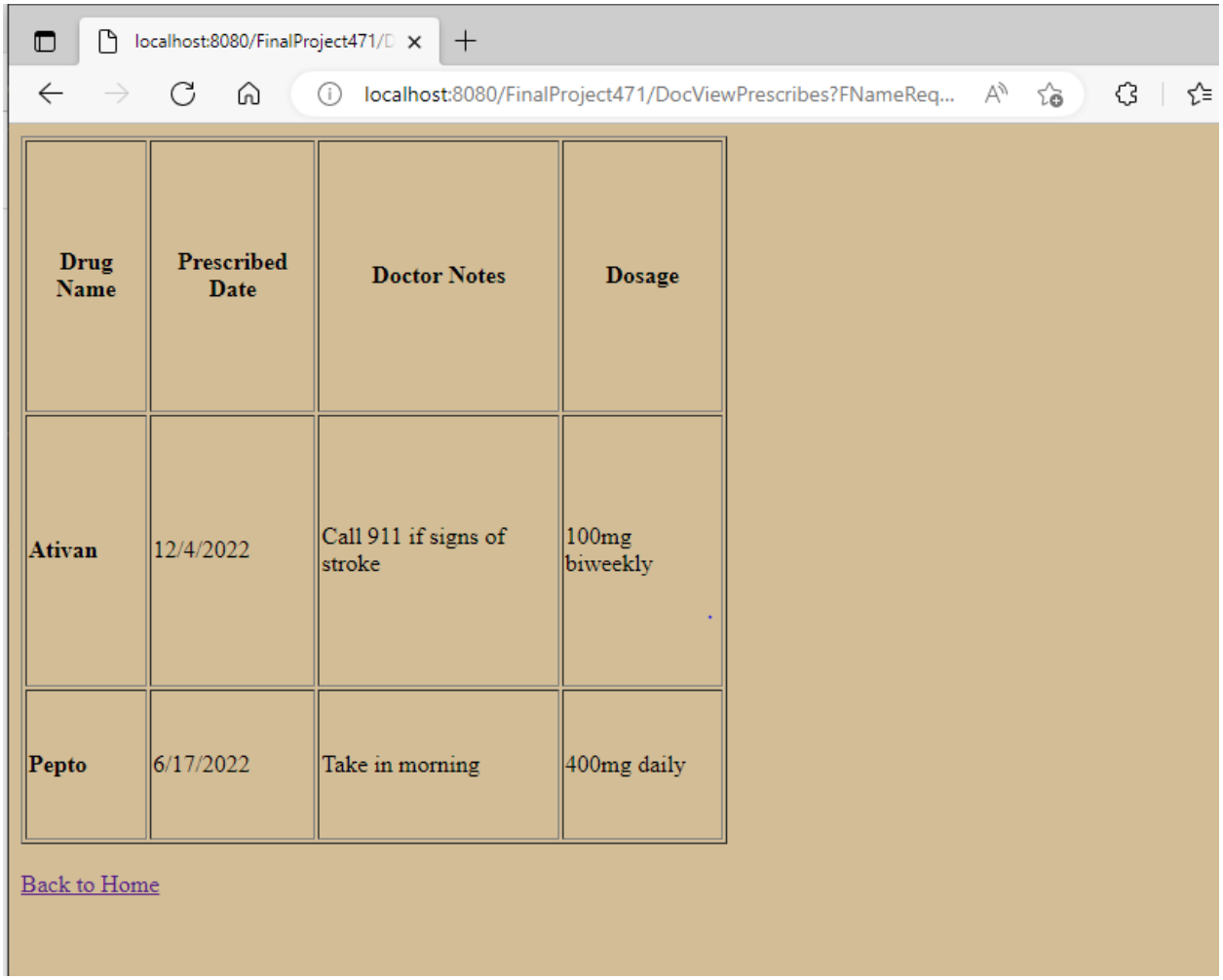


Figure U.48: Result of figure U.47 now

But what if the drug that the doctor wants to prescribe isn't available? In this case, the doctor can simply press the "Request New Drug" link on their homepage, which allows them to request the drug to the system.

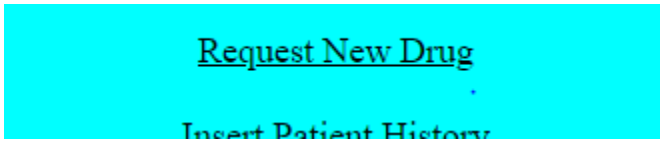


Figure U.49: Request New Drug

Selecting this option:

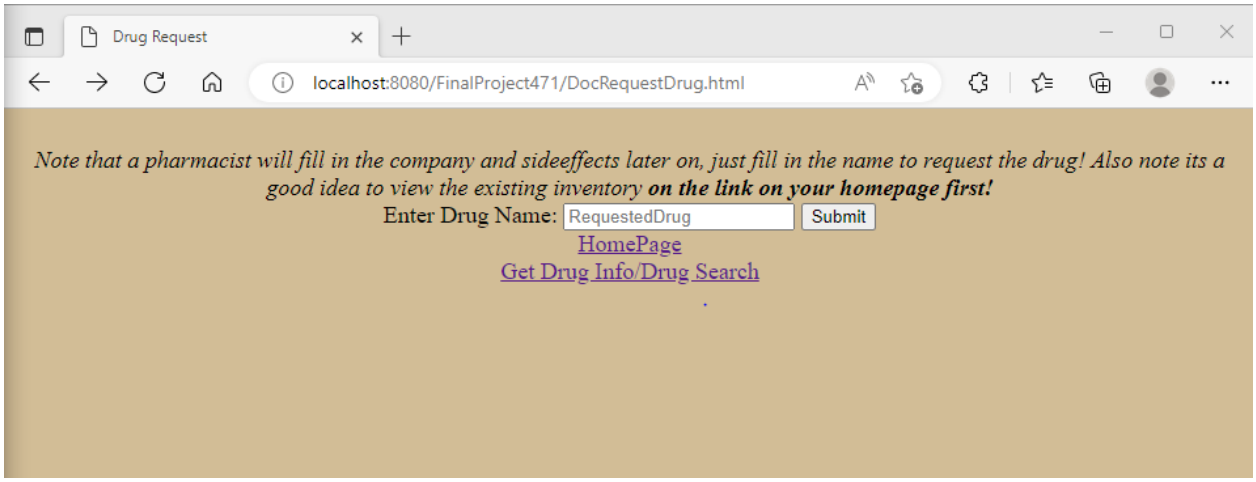


Figure U.50: Selecting Request New Drug. 'Homepage' link brings you back to doctor homepage, 'Get Drug Info/Drug Search' brings you to the drug search (previously discussed), which allows doctors to search if the drug is in the system before requesting it

The functionality of this option is that the doctor can request a drug by name. Doing so sends the drug to the database with NULL values for all non-name attributes. A pharmacist must login and monitor for newly created drugs, and has the capability of seeing newly requested drugs since their last login and it is that pharmacist's job to then fill in the NULL attributes for that requested drug.The pharmacist-side drug request management will be discussed in the pharmacist functionality section of this user manual, and thus we will only focus on the doctor-side for now.

Let's request the drug 'Adderall' to the system.

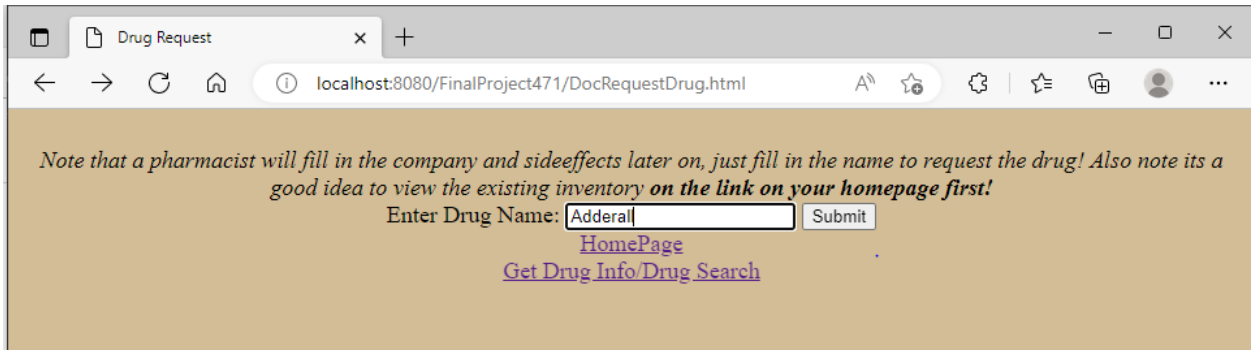


Figure U.51: Doctor requests drug Adderall

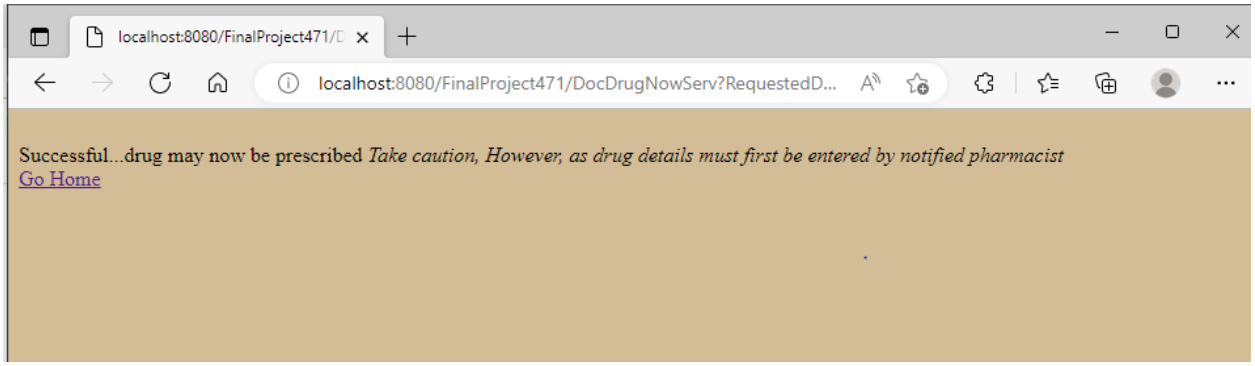


Figure U.52: Result of requesting Adderall

As seen above, the drug has been successfully requested, but requires a pharmacist to fully enter the drug info into the system. However, the drug may still be prescribed in the meantime by doctors. (Refer to Appendix U.7 for database change from this transaction)

Now lets go back and show what happens when we attempt to request an existing drug to the system:

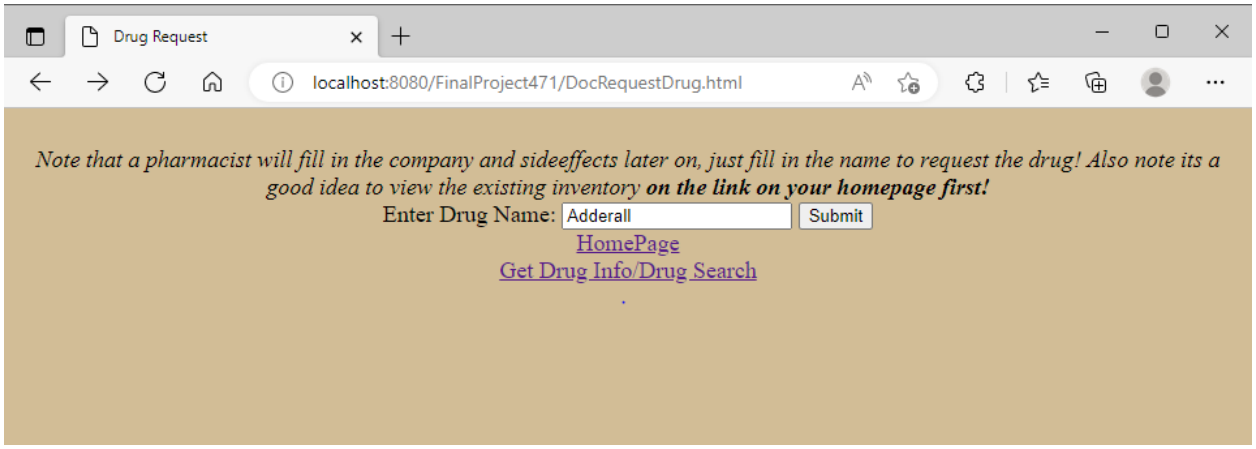


Figure U.53: Re-requesting an existing drug (Adderall again)

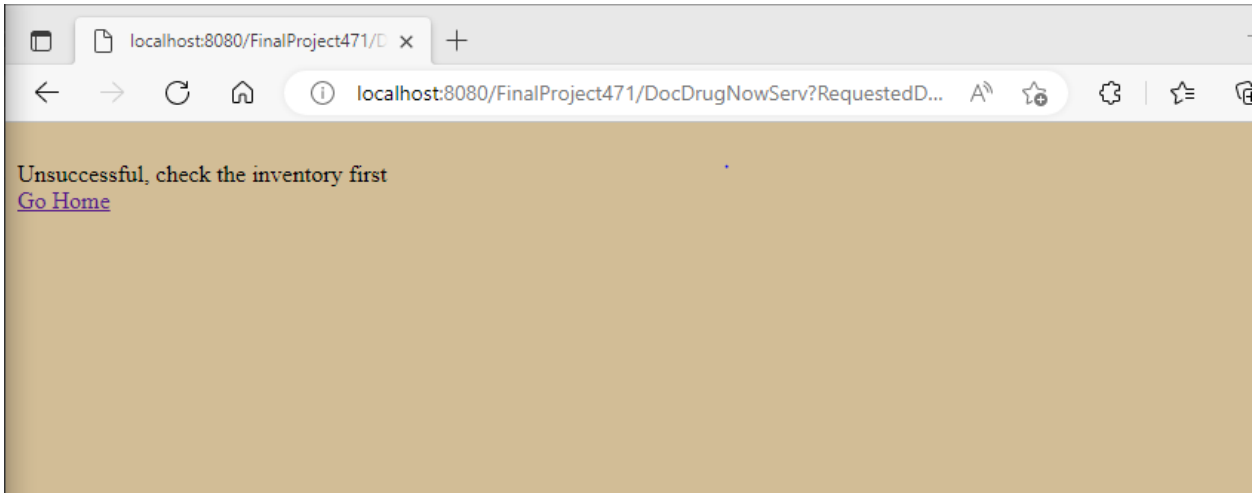


Figure U.54: Message when trying to request an existing drug

As seen above, the system does not allow for the primary key of drugs to be violated and therefore never allows for requesting a duplicate drug.

Now we have seen the drug functionalities of a doctor. We can now move on to the Assign Nurses and Nurse ID Lookup options for a doctor.

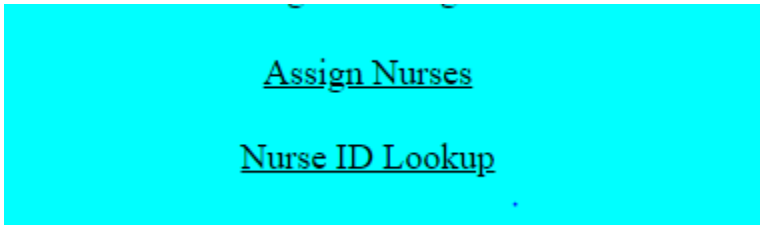


Figure U.55: Assign Nurses and Nurse ID Lookup options on Doctor Homepage

The Assign Nurses option gives doctors the opportunity to manage their staff, specifically the nurses working for them. Firstly, doctors have the capability to, much like with patient, retrieve the unique numeric ID value for a specific nurse, which will be needed for various functions. To do so, the doctor must press Nurse ID Lookup.

Once selected, the search form appears:

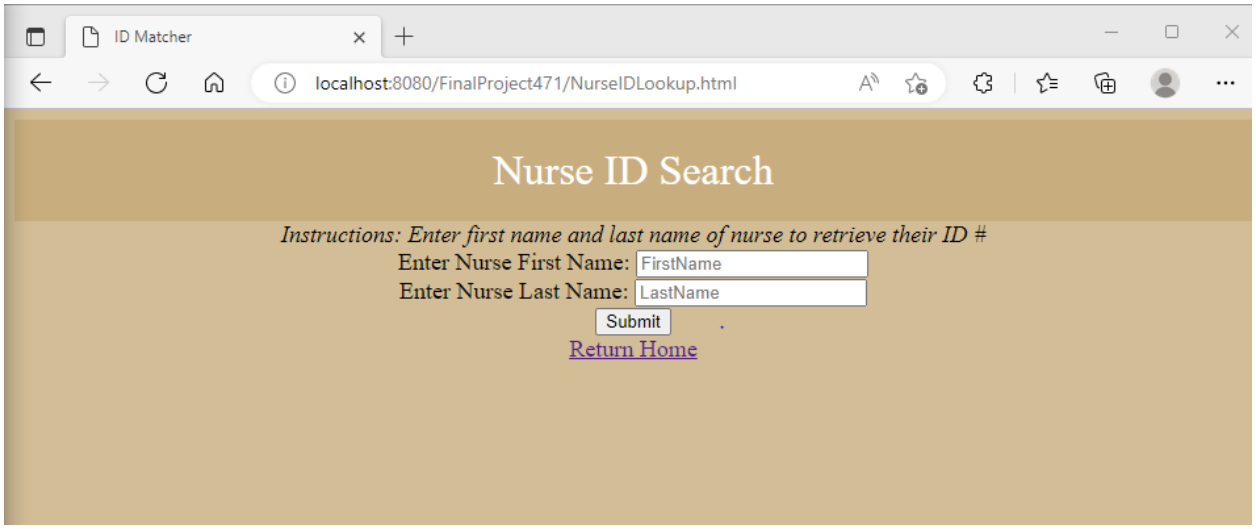


Figure U.56: Nurse ID Search page for Doctors

Here, Doctors must search up a nurse by first name and last name to retrieve ID (refer to the appendix U.1 to see existing nurses in our sample database state). The two possible outcomes

are shown below:

Figure U.57: Searching a non-existent nurse gives a Nurse Does not Exist message, with links to the Doctor Homepage as well as the option to go back to do another search

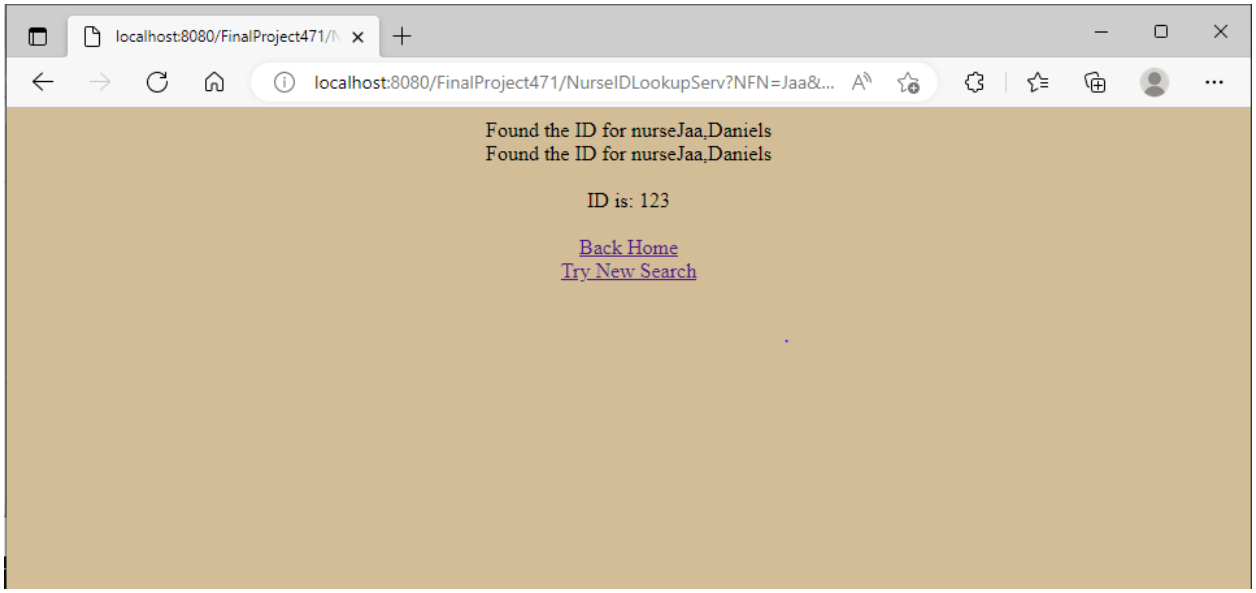


Figure U.58: A successful ID search for Nurse ("Jaa Daniels") (refer to appendix u1 to see nurses in our sample)

Now a doctor can retrieve the ID for any nurse in the system. Going back to the homepage, we can select Assign Nurses to assign a nurse to a clinic under our supervision. To do so, we must

first press Assign Nurses option:

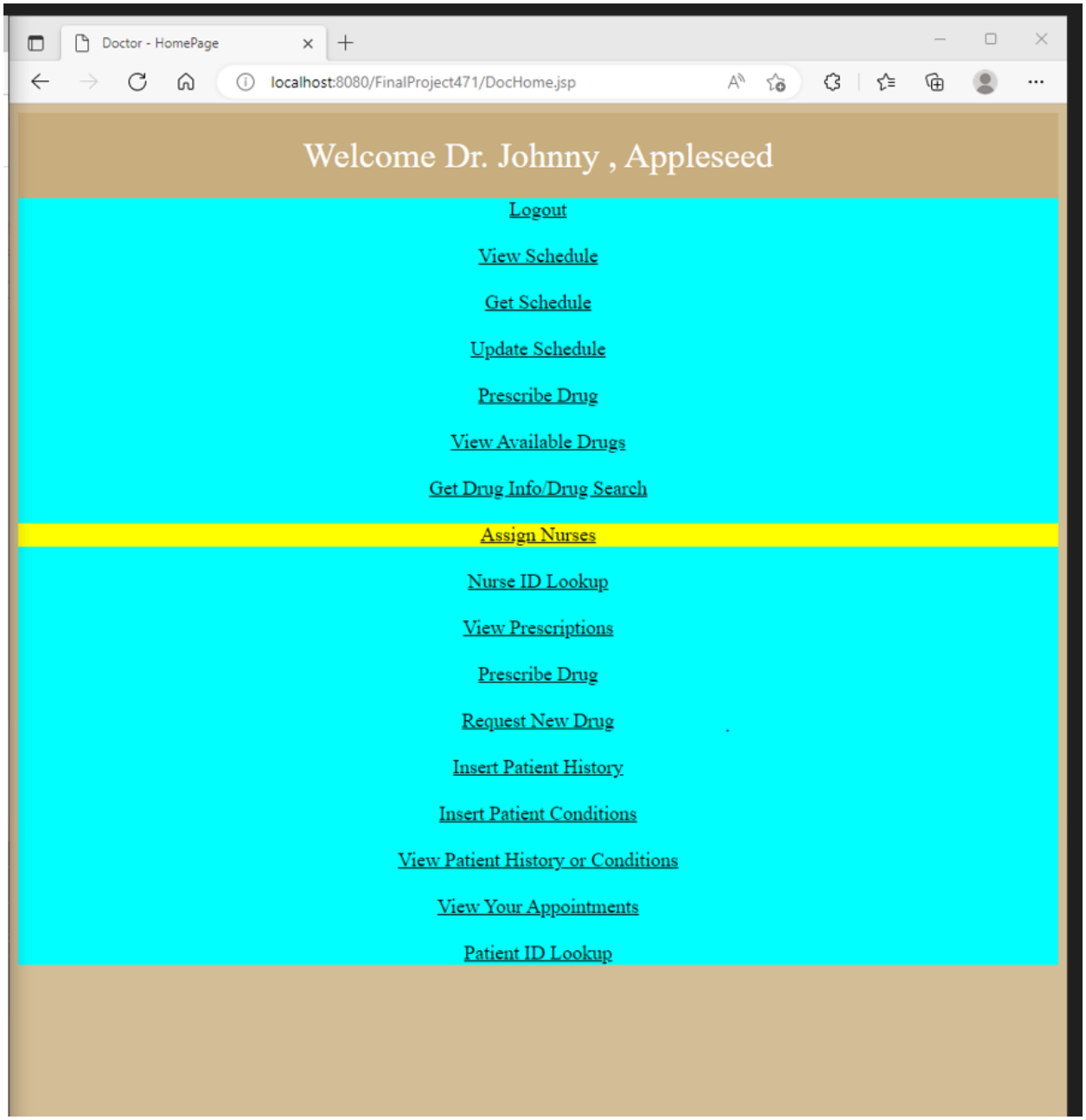


Figure U.59: Selecting Assign Nurses from Doctor Homepage

The first step in assigning a nurse is selecting the clinic:

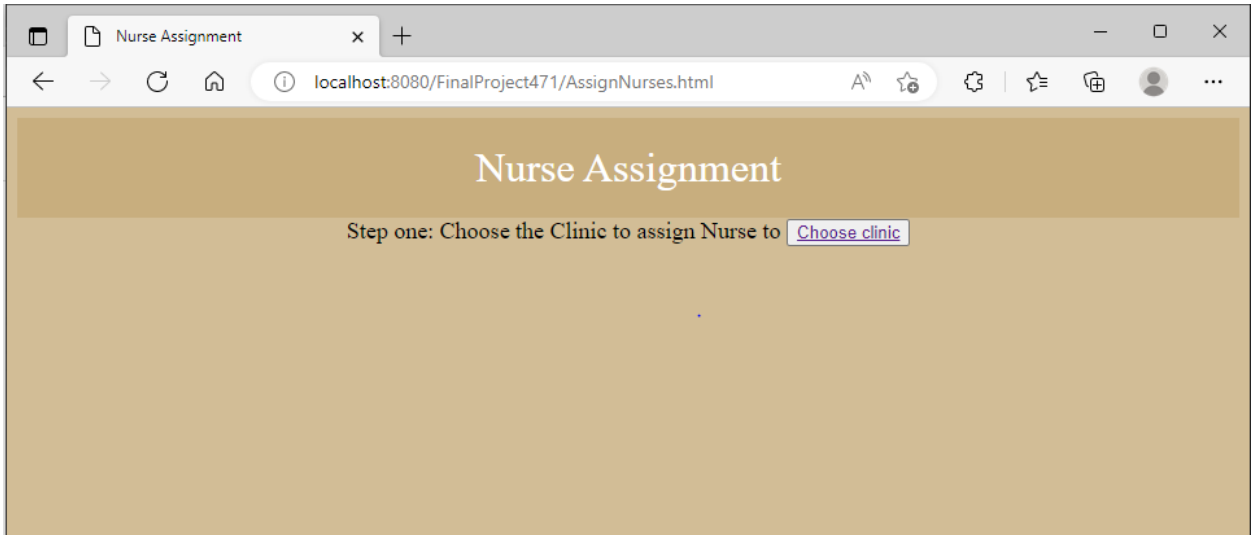


Figure U.60: Step One of Nurse Assignment

Here we see a button urging us to choose a clinic. Once pressed we obtain:

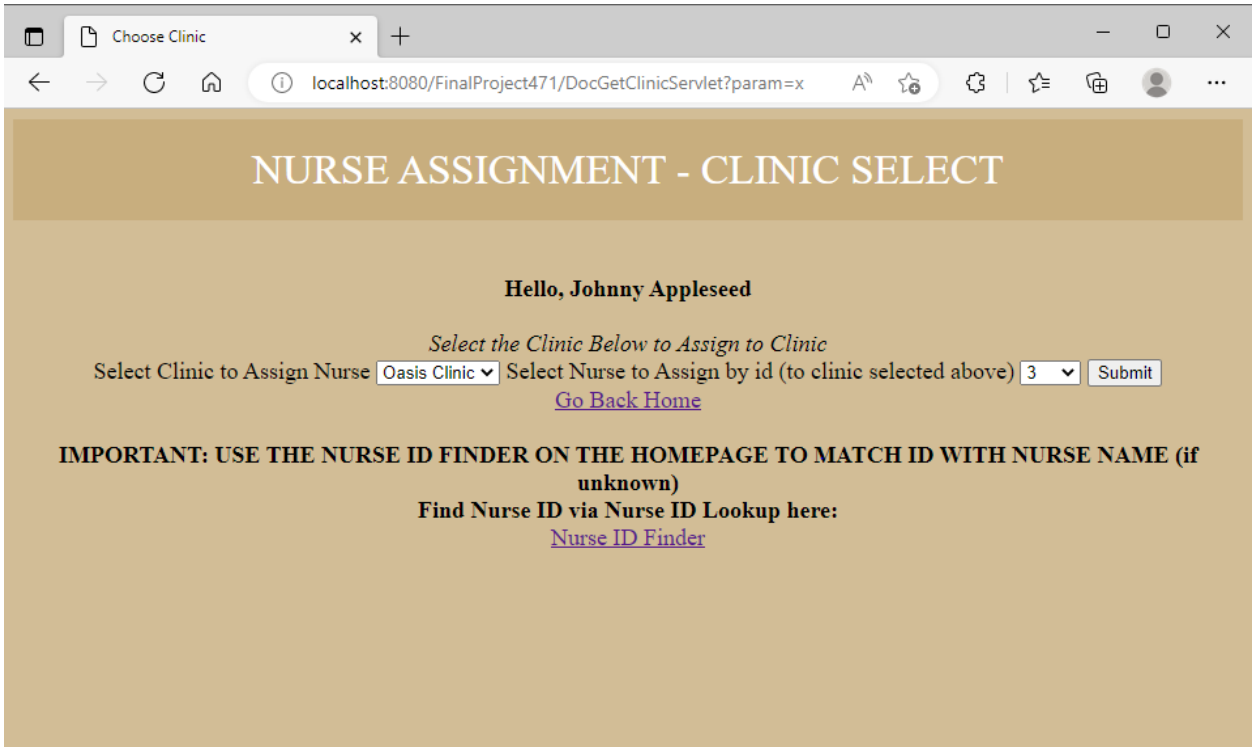


Figure U.61: Next step in nurse assignment (also note the homepage link and nurse id finder link underneath the form)

Now we see a detailed selection page, with a dropdown for clinic (containing every existing clinic as an option), as well as a dropdown menu for each existing Nurse ID (containing every existing nurse).

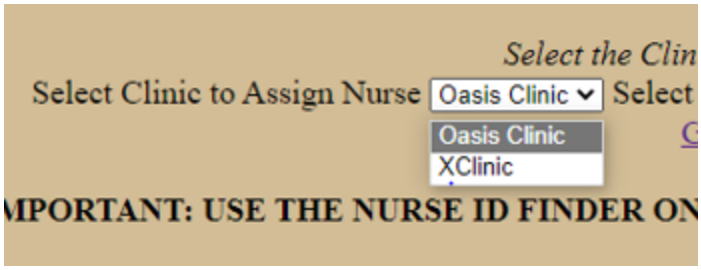


Figure U.61: Clinic option dropdown

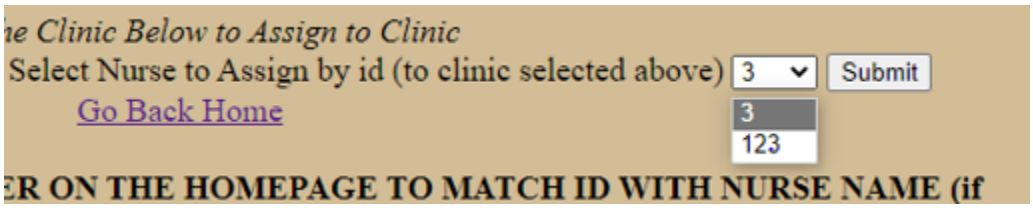


Figure U.62: All nurses dropdown (see appendix u.1 to see all nurses and their IDs)

Because of the quiddity of the dropdown menus, the user is unable to assign a non-existent nurse and/or a non-existent clinic, thus protecting the system.
For this example, lets assign the nurse ‘Ri Lu’ , whose ID equals 3 to the clinic ‘XClinic’. Note that this nurse is currently assigned to Oasis Clinic (Appendix U.8 for full database transactions).

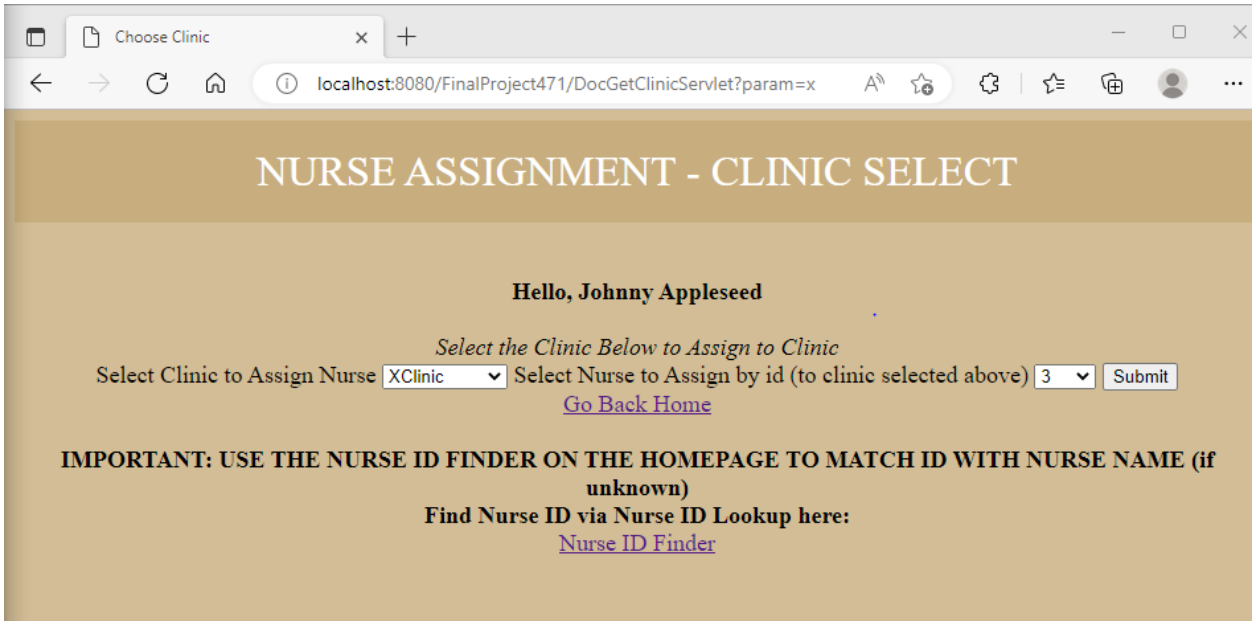


Figure U.63: Assigning nurse Ri Lu

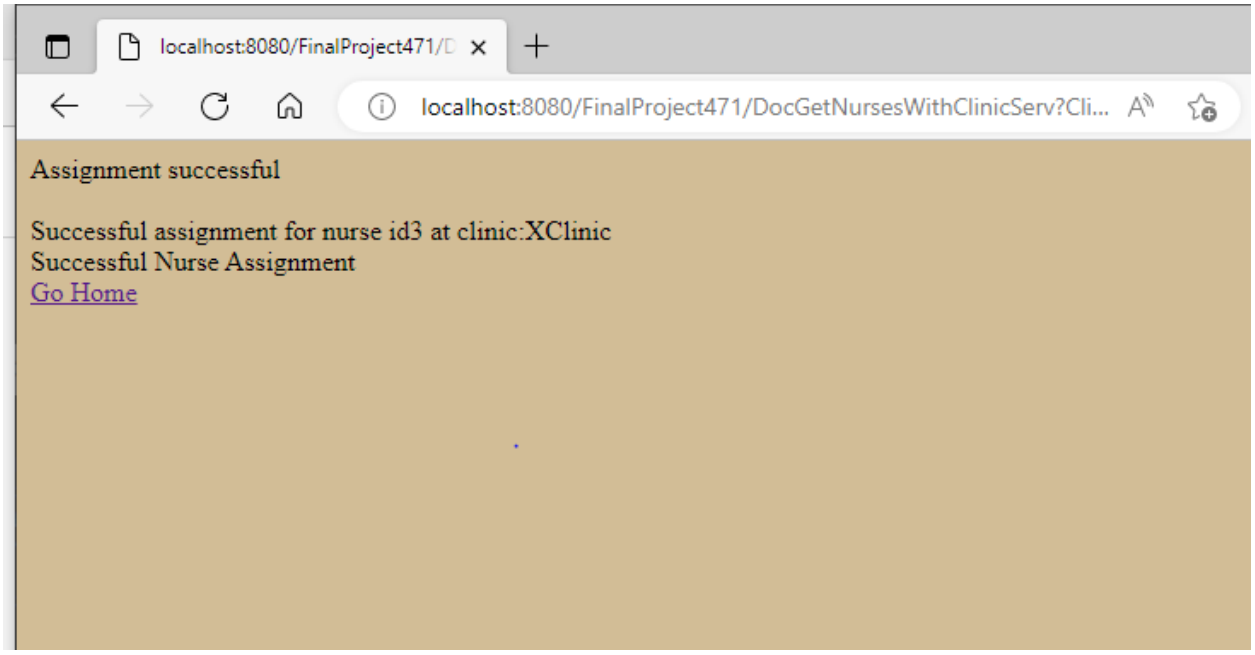


Figure U.64: Successful assignment of Ri Lu

As we can see, we get a successful assignment message, with a link to the doctor homepage. Refer to appendix u.8 to view the database change.
Now, lets go back and see what happens when we change the same nurse by attempting to assign them to their existing clinic (ie. try to assign them to XClinic which they are now assigned to).
Repeating the previous steps, we select the 'assign nurse' option and fill in the form for ID=3 and Clinic='XClinic'. Submitting we see:

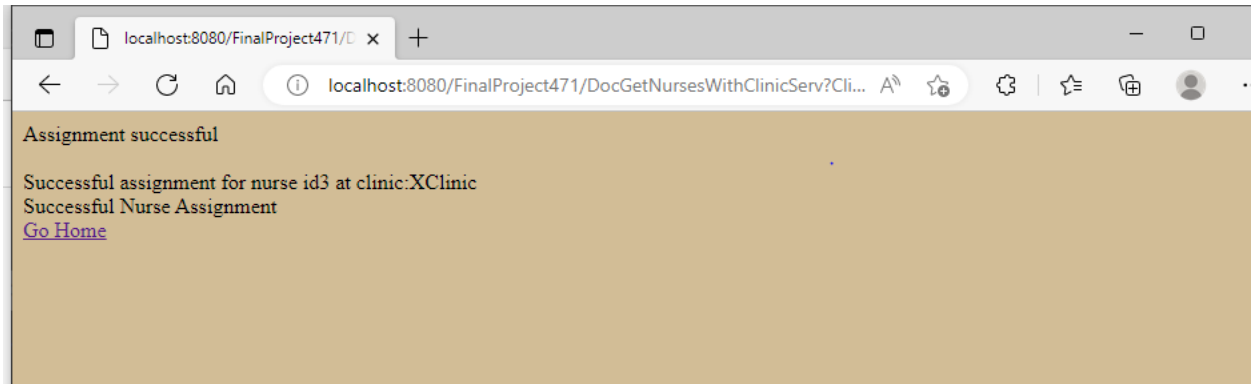


Figure U.65: Assigning RI Lu to their existing clinic

We see the successful assignment message again, as we have assigned them from Xclinic to Xclinic. Although the nature of this change is meaningless, it is nonetheless a tangible and valid assignment.

Now we can discuss the few remaining doctor functions available on their homepage: Insert patient history, insert patient conditions, view patient history or conditions and view appointments.

First, lets show the view appointments option:

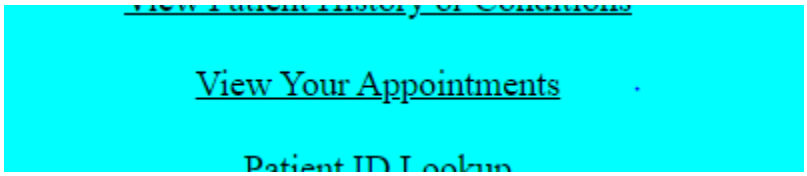


Figure U.66: View Your Appointments option

Here, a doctor can view any scheduled appointments that a patient has booked in the system. First, lets look at Dr. Appleseed's, which is empty (since we just created him):

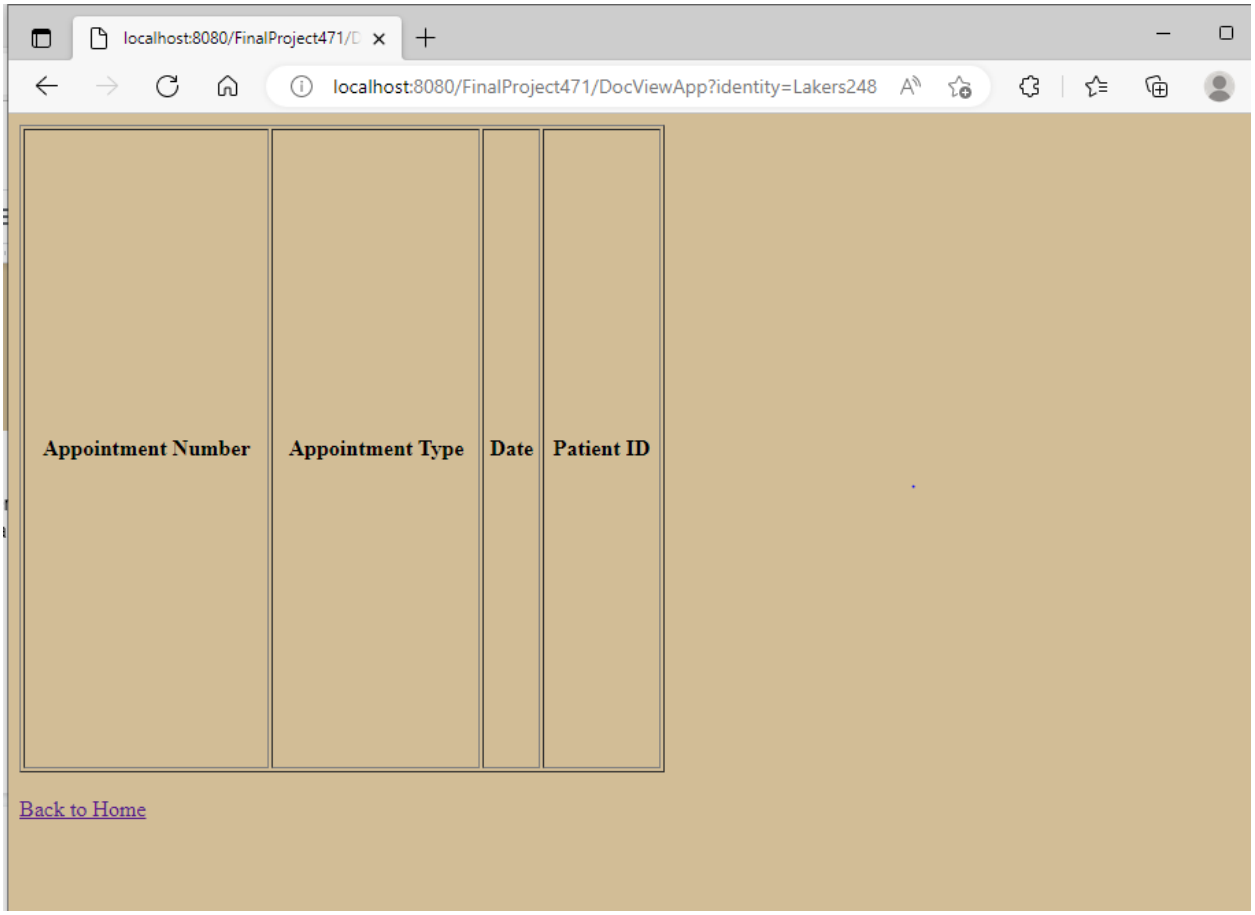


Figure U.67: An empty view appointment table for a doctor with no scheduled appointments

Now, lets briefly logout of Dr.Appleseed's account and login to Dr.Fir Nam to see what a doctor with appointments booked looks like for this option (refer to appendix u.1 to see that this doctor has an appointment).

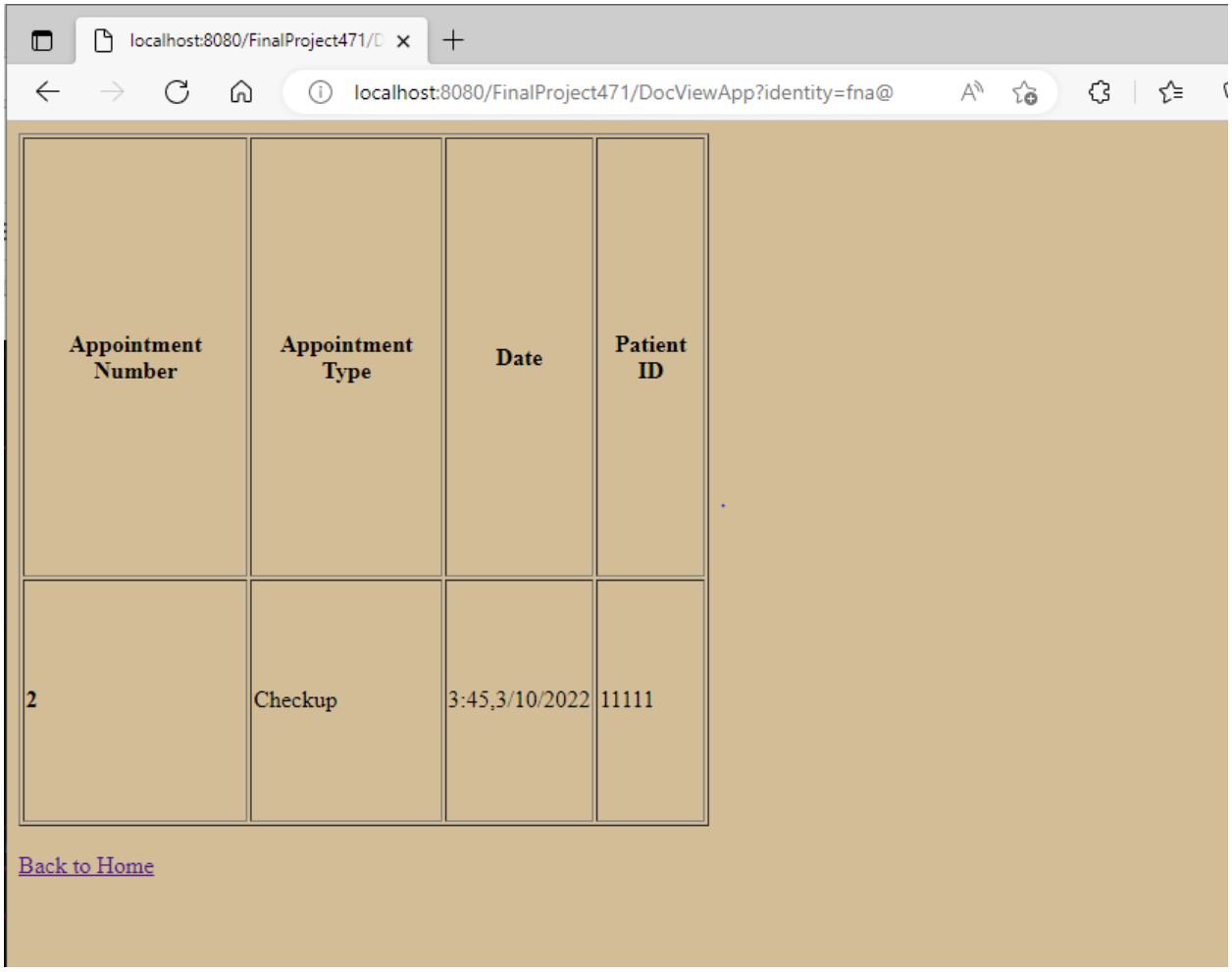


Figure U.68: Dr.Fir Nam's view appointments - which is not empty

Now lets logout and log back in to Dr.Appleseed's account to continue along with our example.

Now we can discuss Insert Patient History.

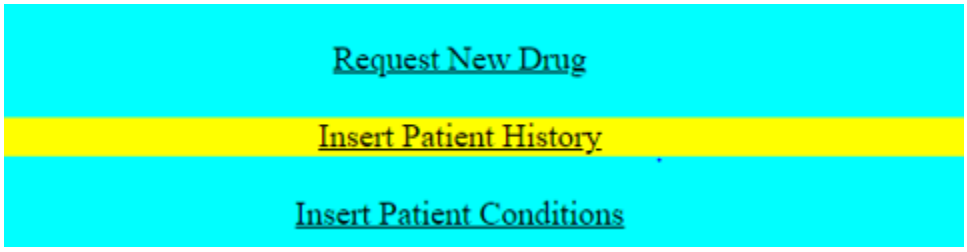


Figure U.69: Insert Patient History

Here, a doctor has the ability to insert history into a patient's records. The policy of our system is to (a) Insert a new record if no history exists for that patient, and (b) Concatenate the history to the existing history if records already exist. This is to maintain the historical accuracy and integrity of a patient's medical history.

Pressing this option yields this option:

Figure U.70: The Insert History page for Doctors

Here, a doctor must enter the first name and last name for a patient, as well as the history to add.
First, lets enter history for a nonsensical patient that does not exist:

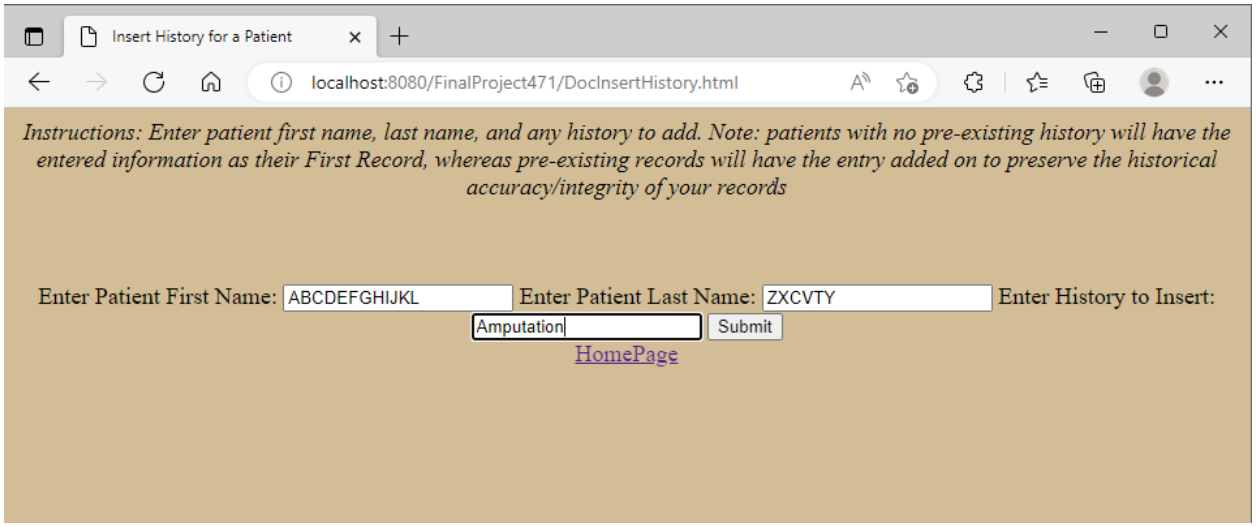


Figure U.71: Trying to insert history to non-existent patient

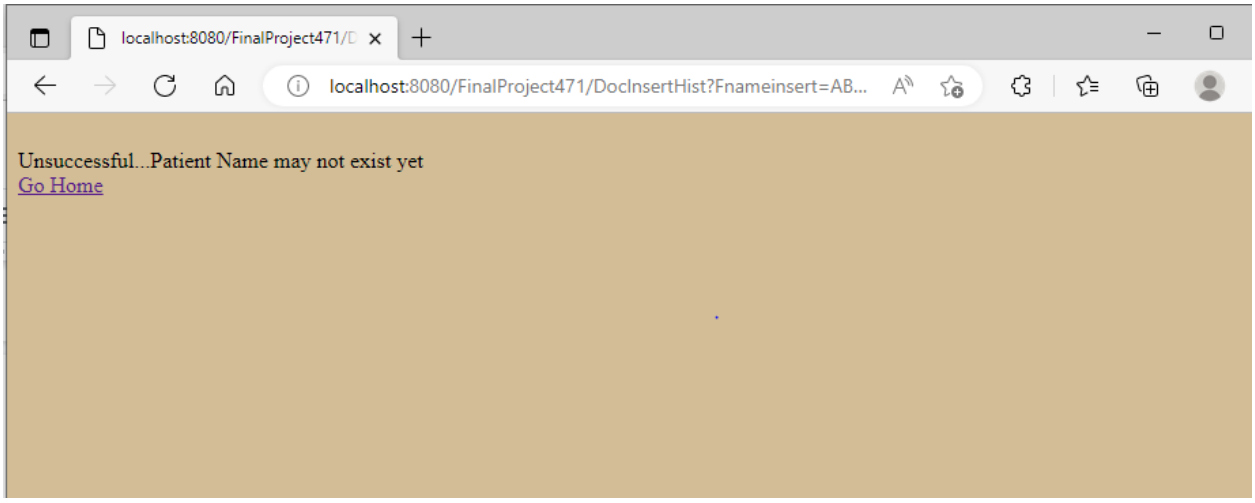


Figure U.72: The resulting message from figure u.71 above

Now lets try to add a historical record to a patient with none. (Refer to appendix u.9 to see the database changes here and the following steps).
Going home and re-selecting insert patient history, lets add a history to a patient with no pre-existing history, eg. patient 462380, Frank Ocean.

Figure U.73: Inserting medical history

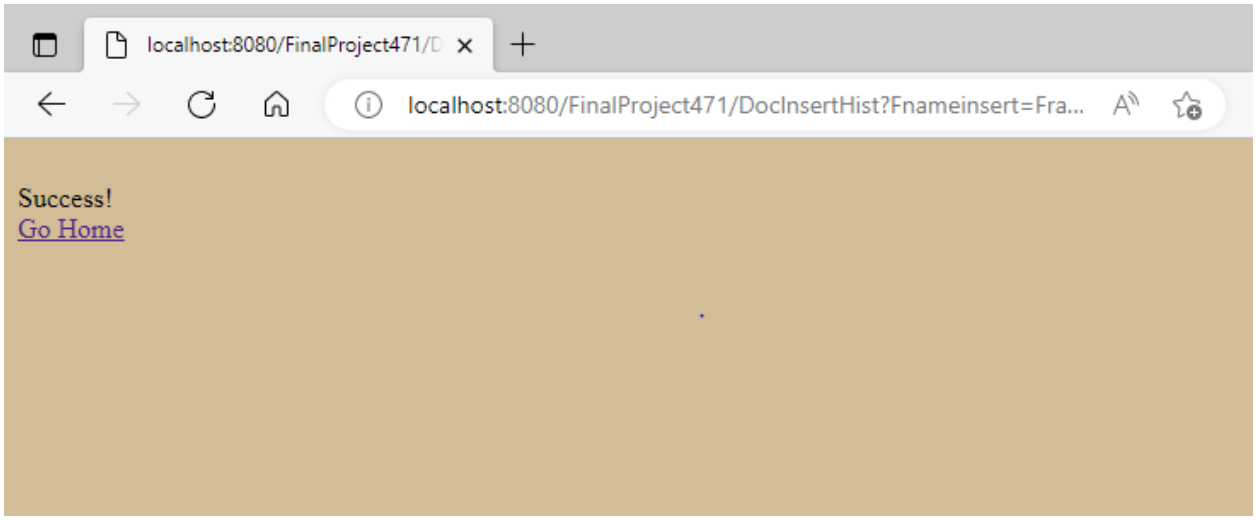


Figure U.74: Result from above

Now we see a success message, and we can see in appendix u.9.2 the change to the database.

Now lets add another history record to the same patient and see how the system concatenates the records rather than overwrites.

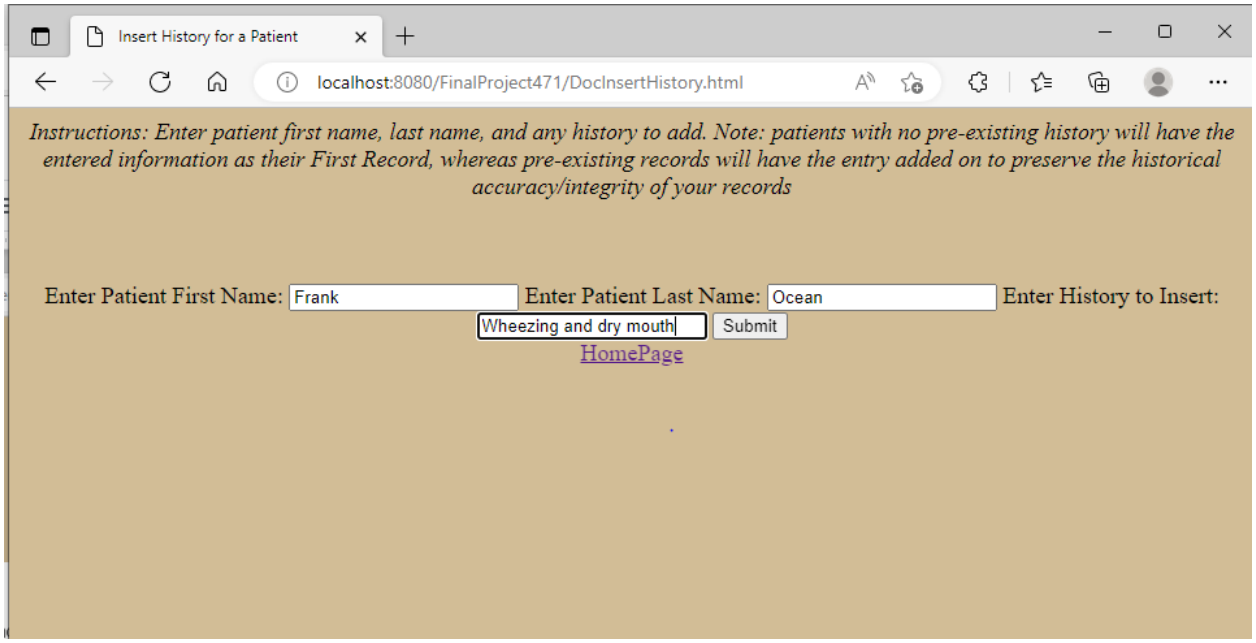


Figure U.75: Adding another record to the same patient

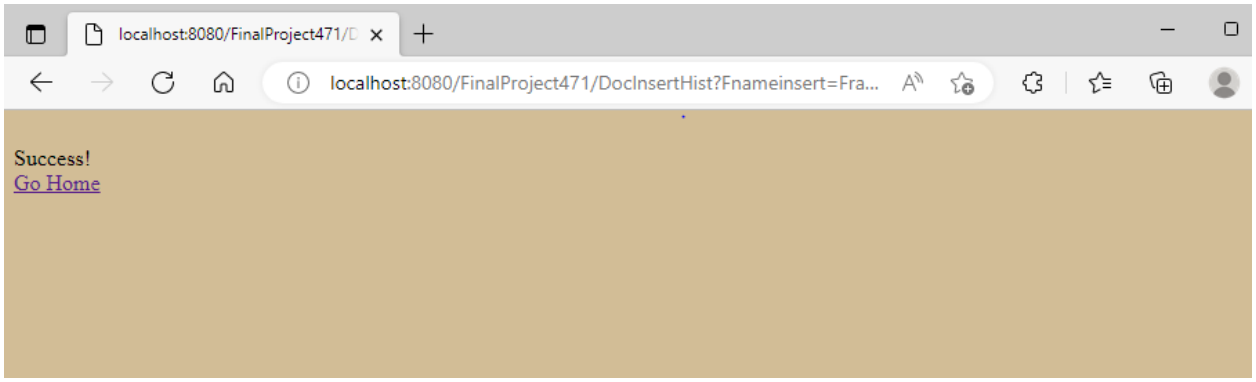


Figure U.76: The resulting message from above

In order to see these changes, we need to show the 'View Patient history or conditions' option.

Now we will go home and press the View Patient History or conditions option. Presing this gives us this page:

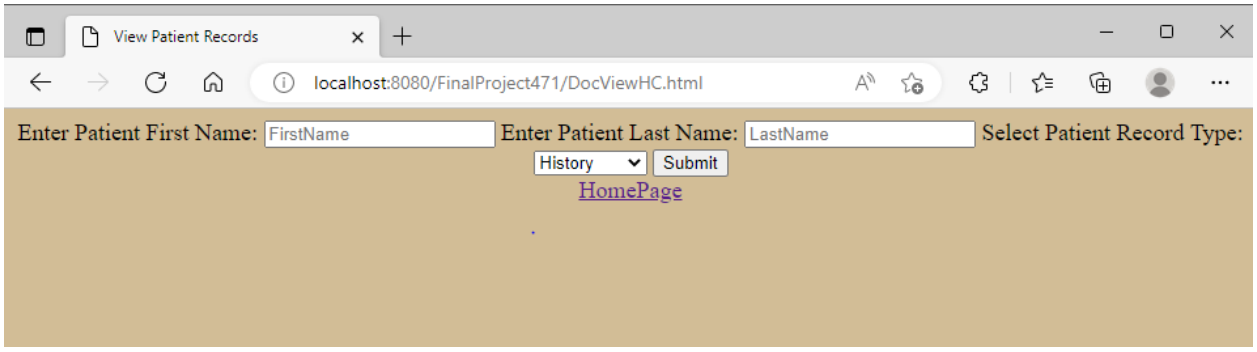


Figure U.77: View Patient History or Conditions option

Here, we can search a patient by name, and choose to see (i)history, or (ii)conditions for that patient.

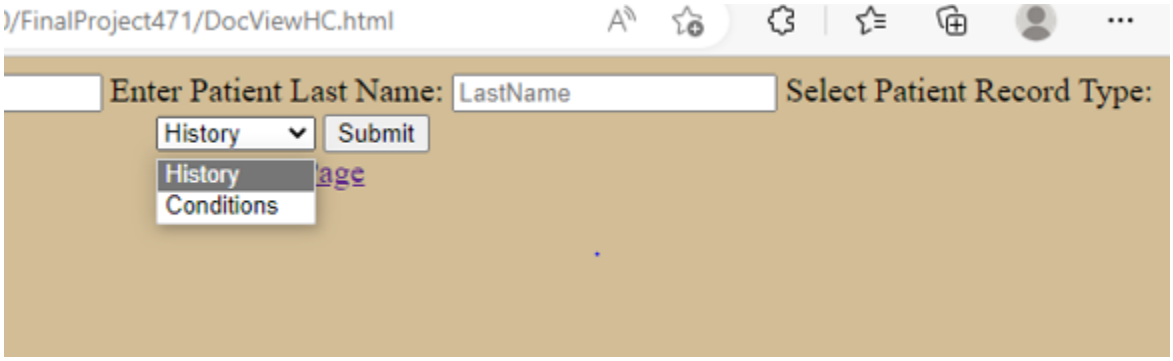


Figure U.78: View records options - History or Conditions

Lets first select history for patient Frank Ocean, who we have manipulated their records in the previous transactions.
Entering Frank, Ocean and selecting History yields this result:

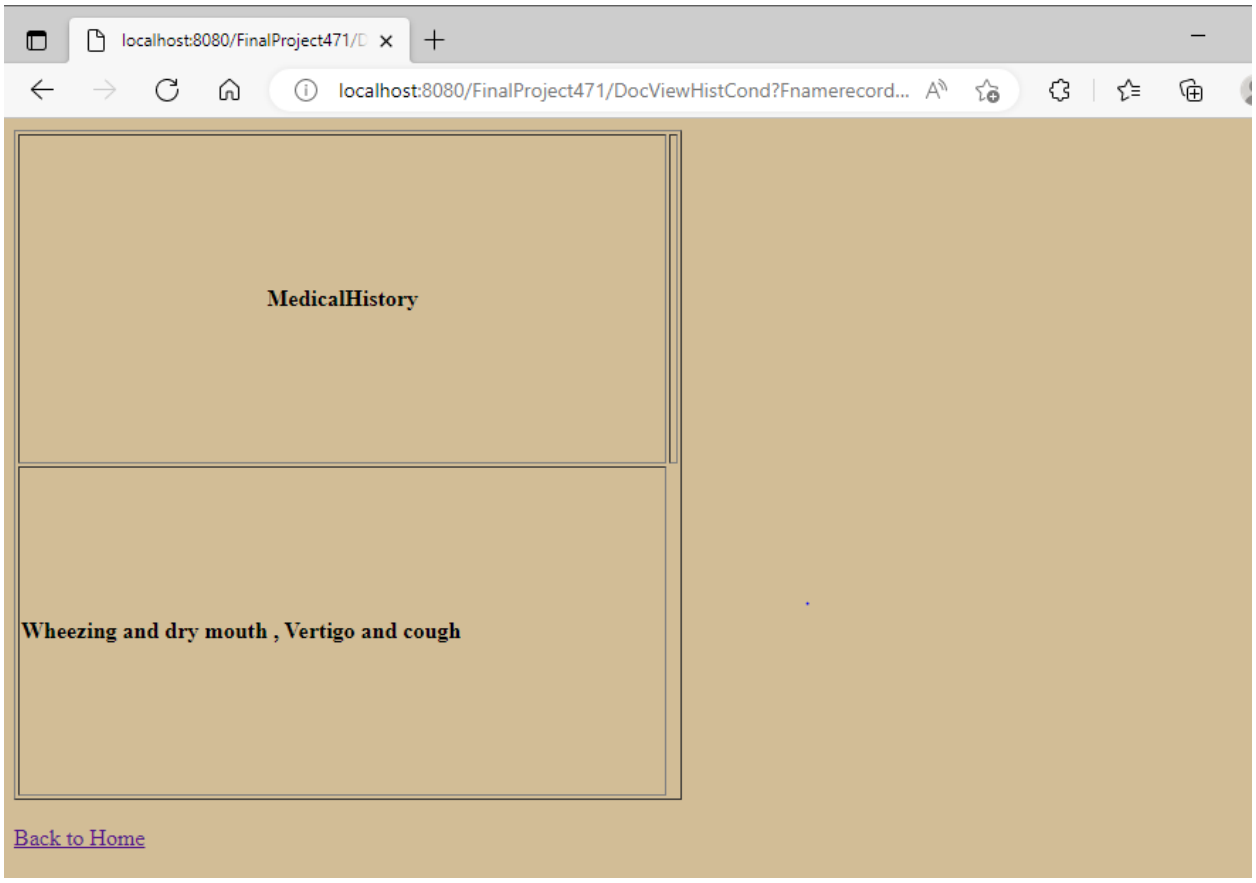


Figure U.79: Medical History retrieved for Frank Ocean. Note that records are concatenated with a comma rather than overwritten

Now lets do this again for a patient that does not exist:

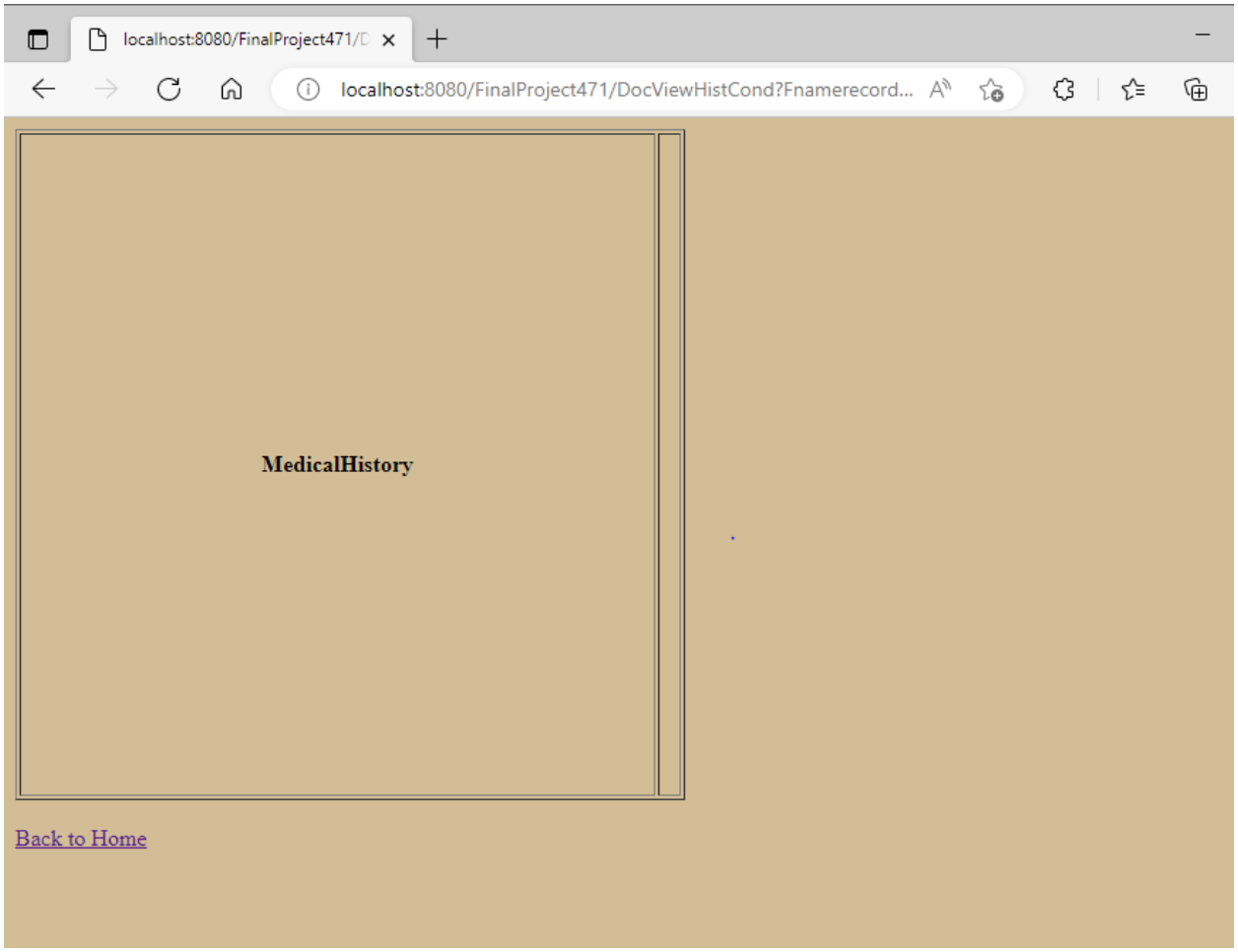


Figure U.80: Empty medical history for non-existent patient

Finally, we will go back and select conditions instead, but for Patient Pat Zero, since they have conditions (whereas Frank Ocean would simply print out an empty table as in figure u.80).

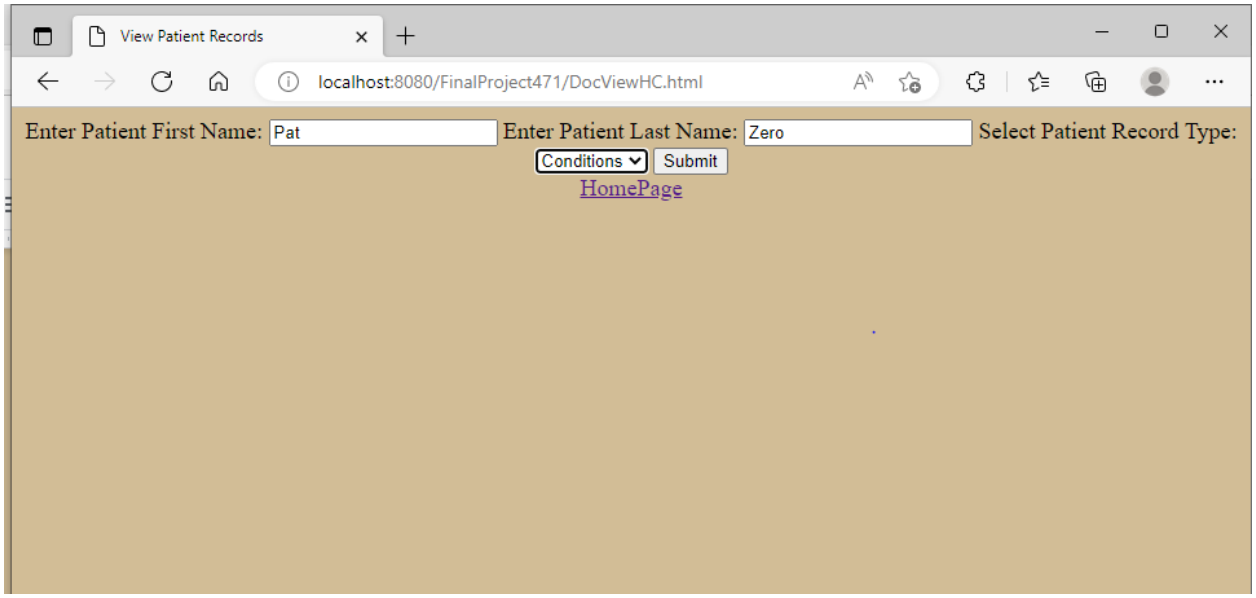


Figure U.81: Selecting to view conditions for patient Pat Zero, who has records

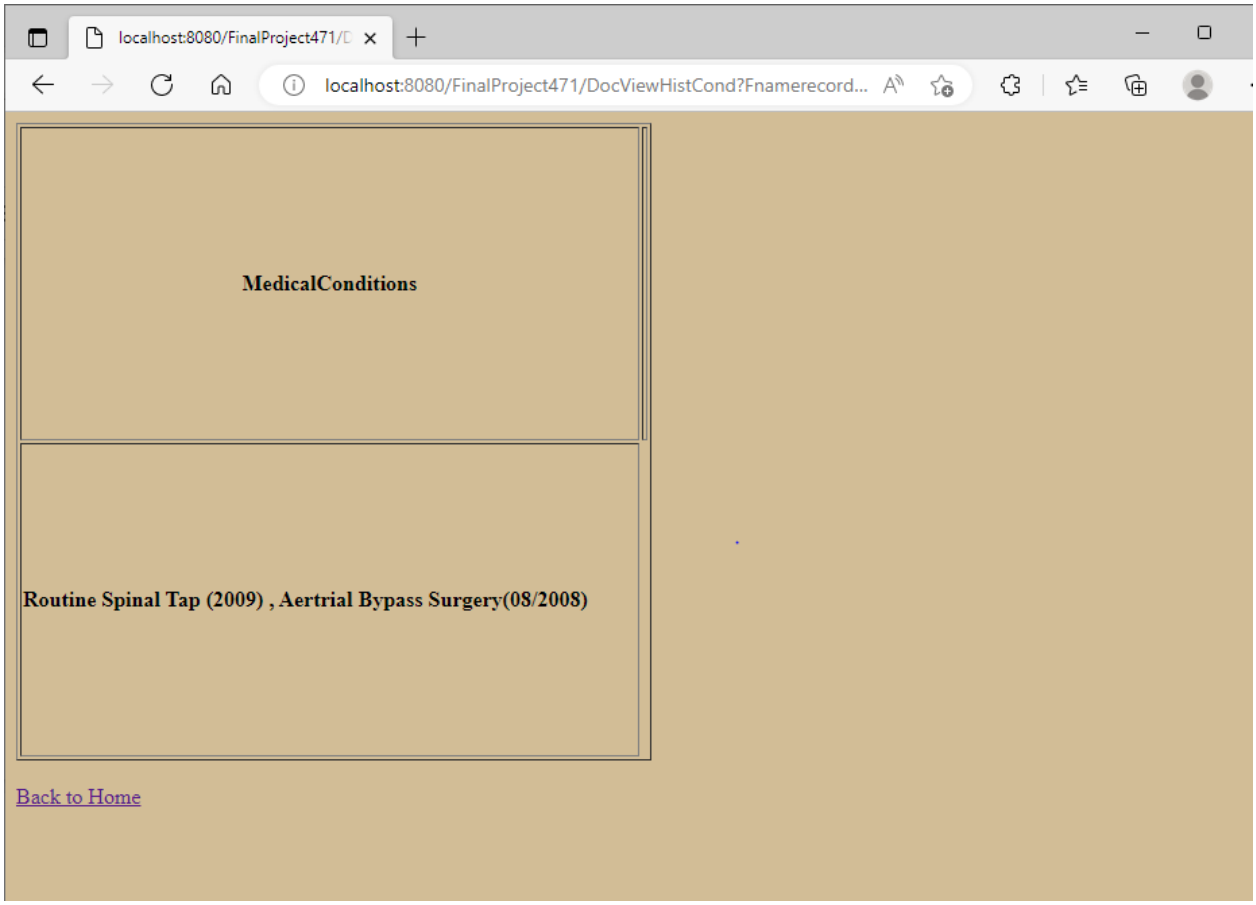


Figure U.82: The result of figure U.81

As we can see, the conditions are printed out for that patient.

Finally, we can discuss the insert patient conditions, which is very similar to the insert patient medical history. In fact, we have just seen how to print out patient conditions above.

Firstly, the policy is the same for patient conditions, where existing records are concatenated rather than overwritten.
Now lets select insert patient conditions from the homepage

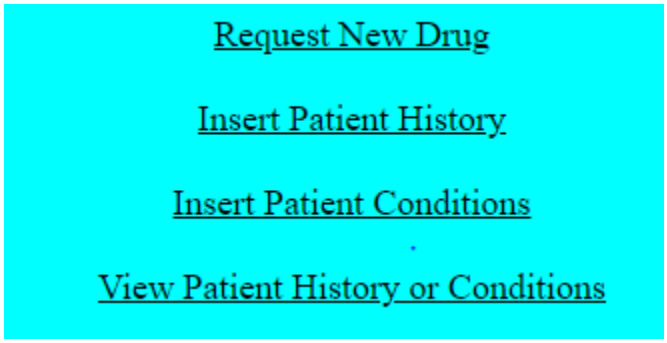


Figure U.83: insert conditions option

Now we have this page:

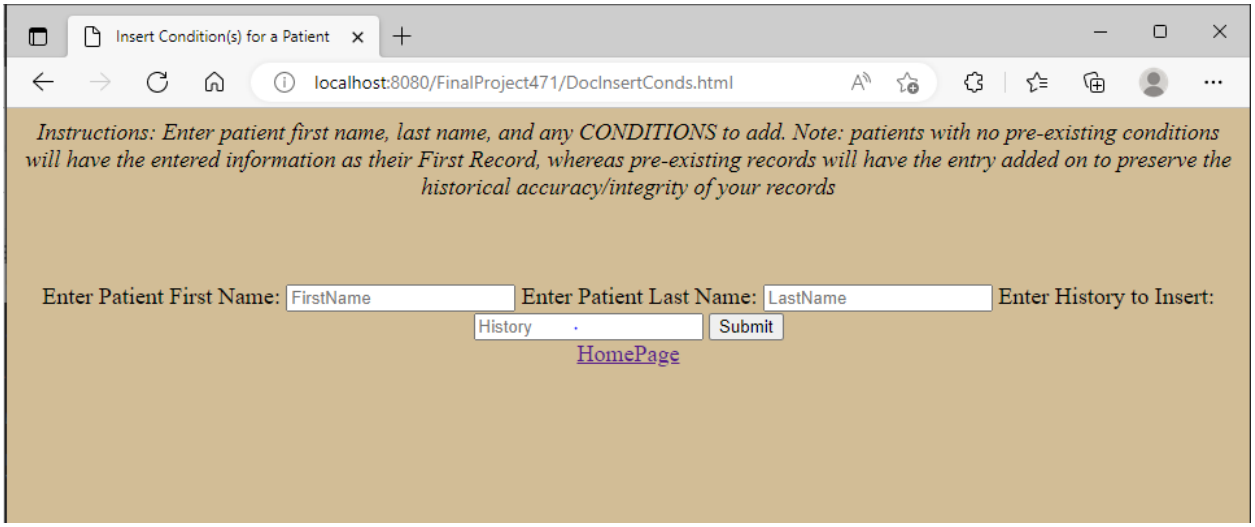


Figure U.84: Insert conditions page

Now lets again fill in the form for Frank Ocean, who is a patient without any conditions yet (see appendix u.10 to track these transactions) .

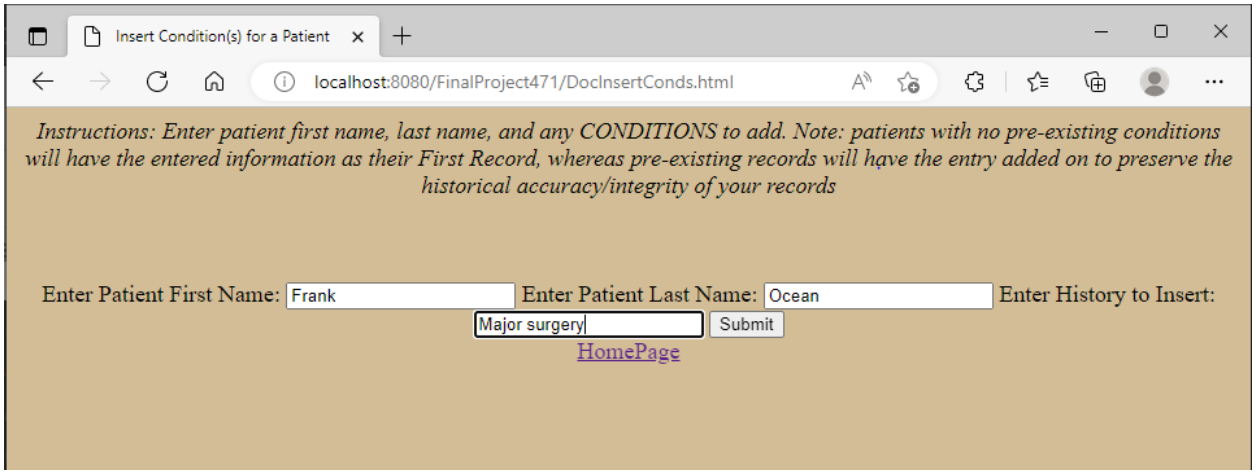


Figure U.85: Entering conditions to a patient who has no existing records

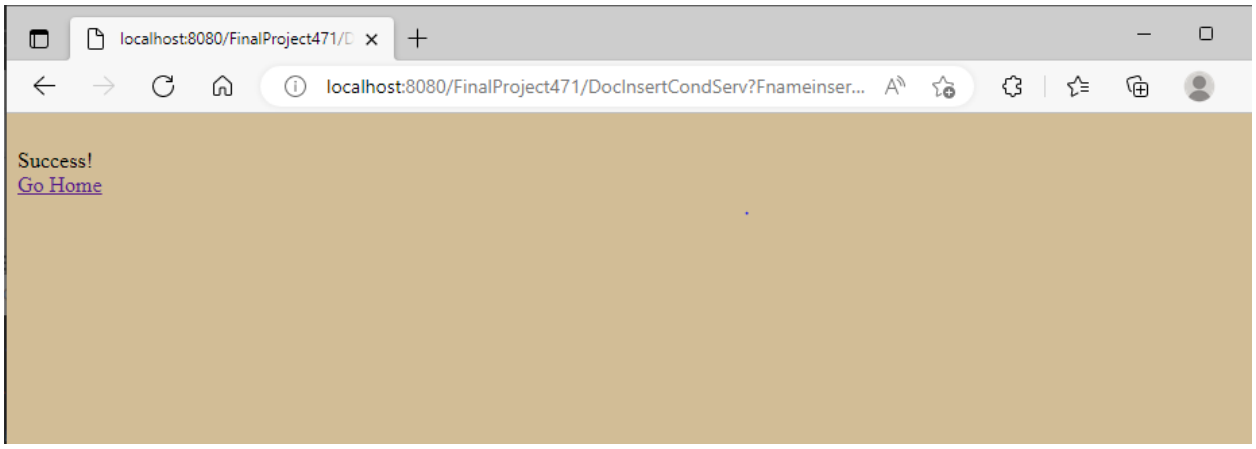


Figure U.86: The resulting message

Now lets try adding another condition to the same patient:

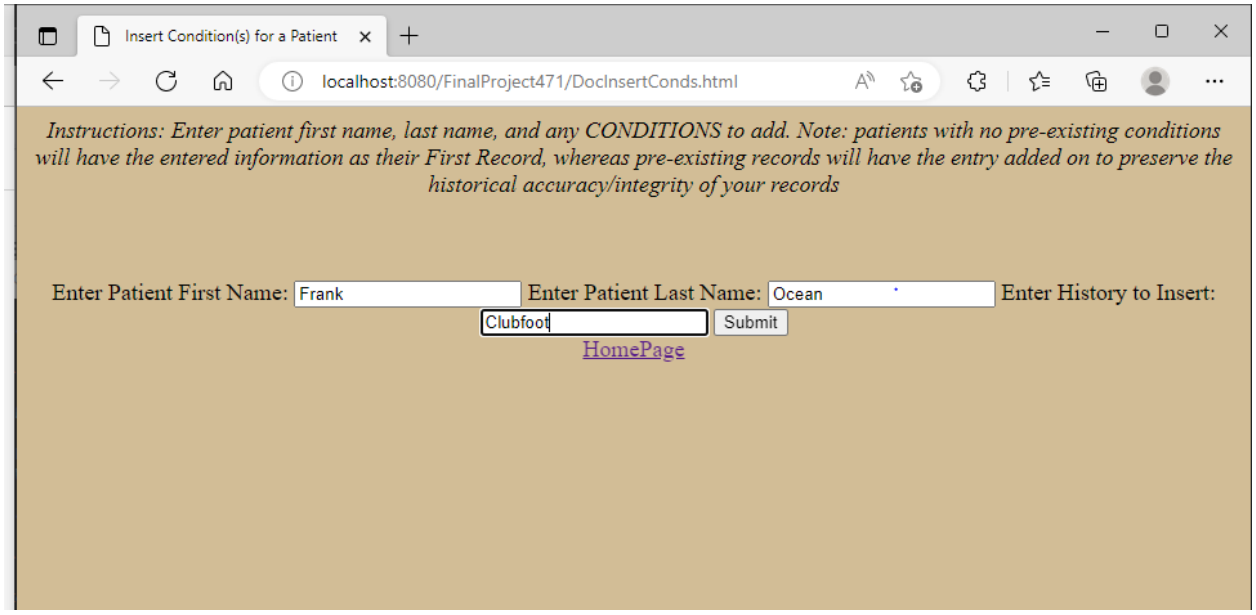


Figure U.87: Inserting another condition to the same patient

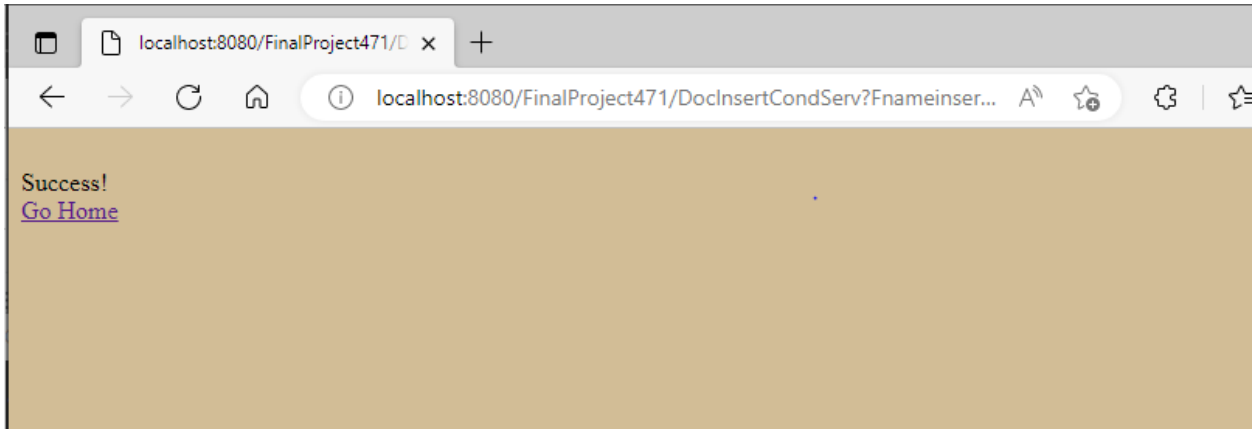


Figure U.88: Successful transaction

The transaction record can be checked in appendix u.10. (Note that the condition is concatenated rather than overwritten).

Finally, we can attempt to insert conditions to a patient that does not exist:

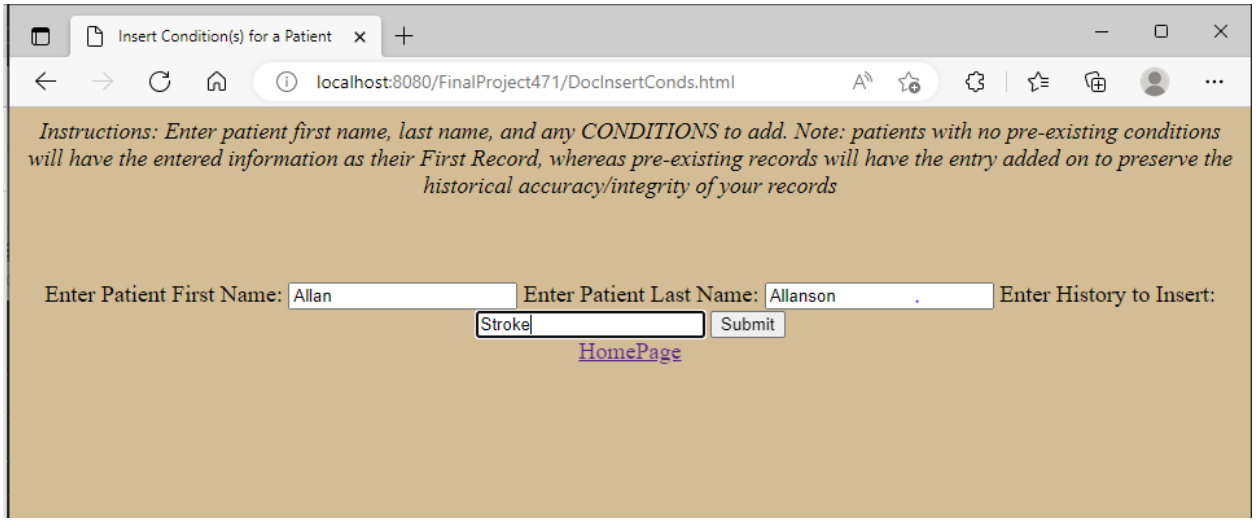


Figure U.89: Trying to insert a condition to a patient who does not exist

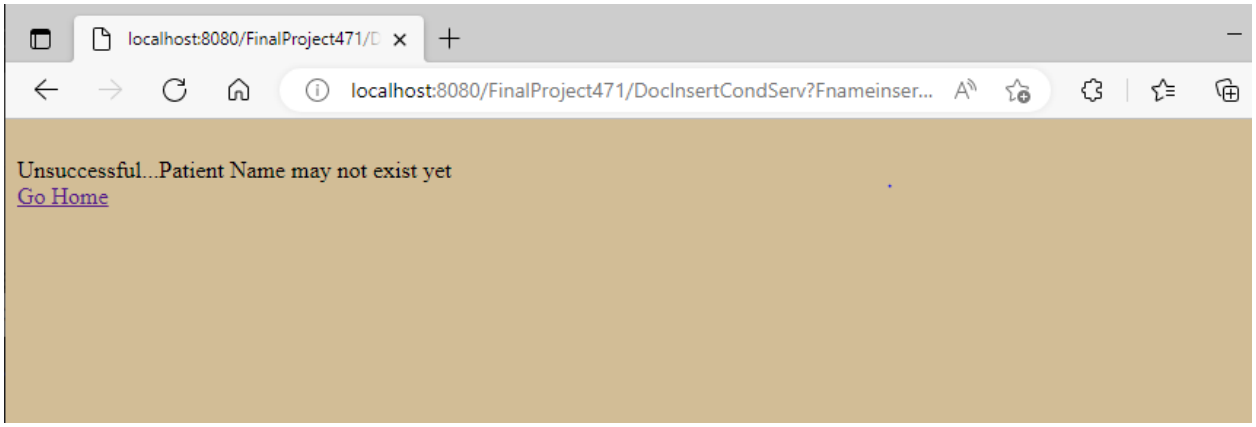


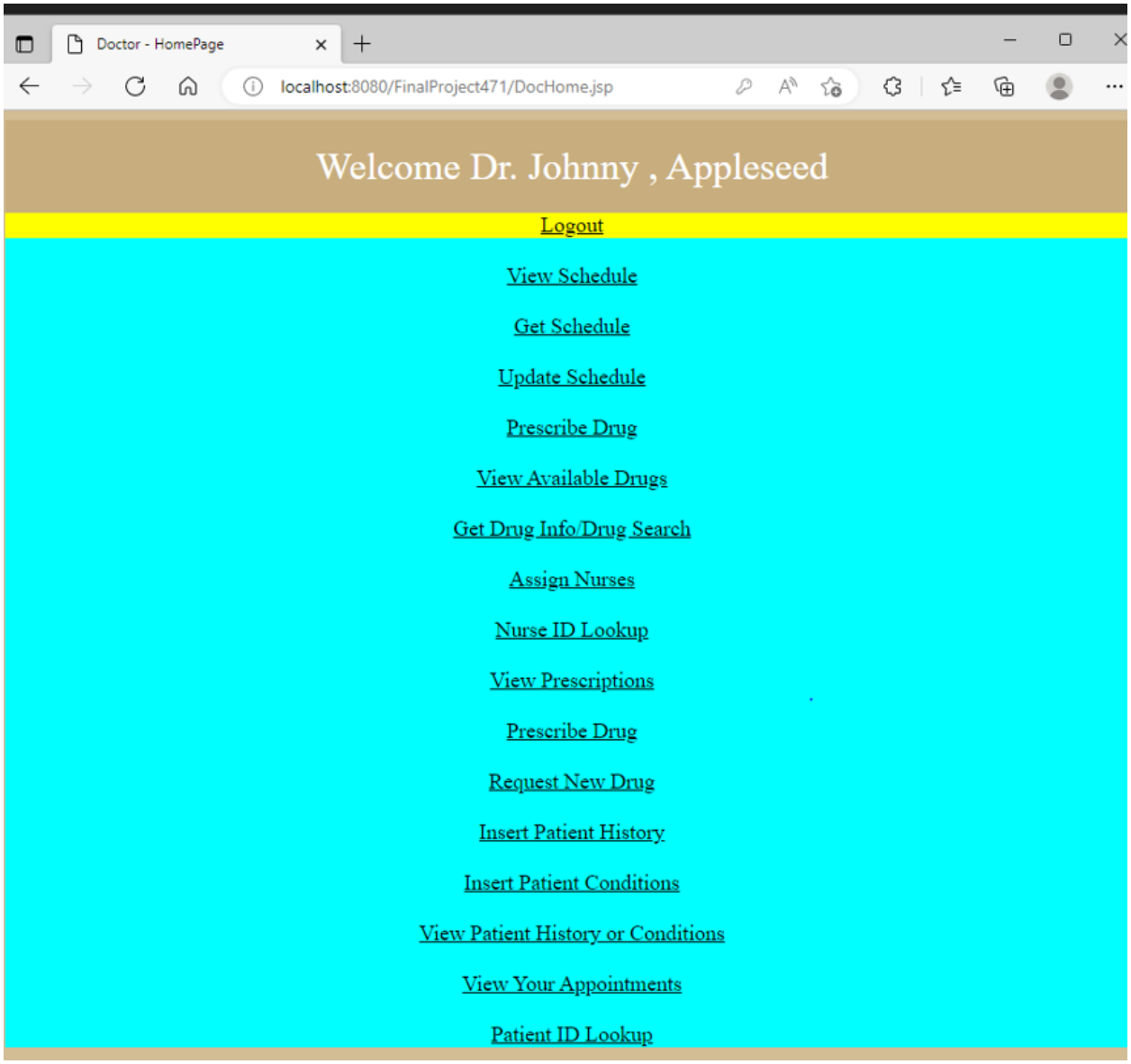
Figure U.90: The resulting message

As shown, the record is not inserted due to the patient not being existent.

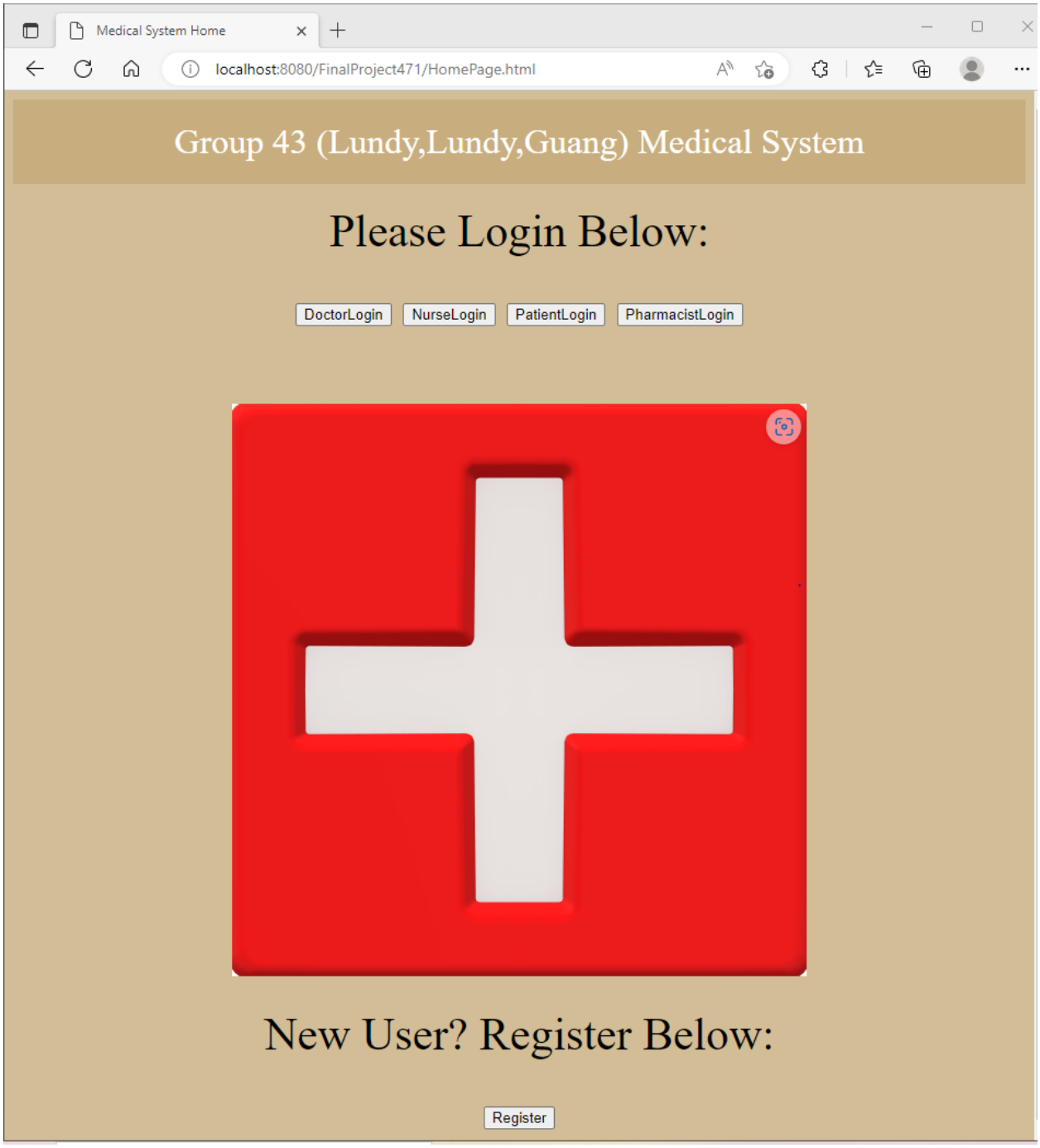
That concludes the discussion of the doctor functionality. Next, we will go over the pharmacist user functionality.

Pharmacist Functionalities

To begin with the pharmacist functionalities, let us first register as a new Pharmacist.
First, lets make sure we press logout from the doctor homepage to ensure we are back at the medical system homepage.



Logging out from doctor

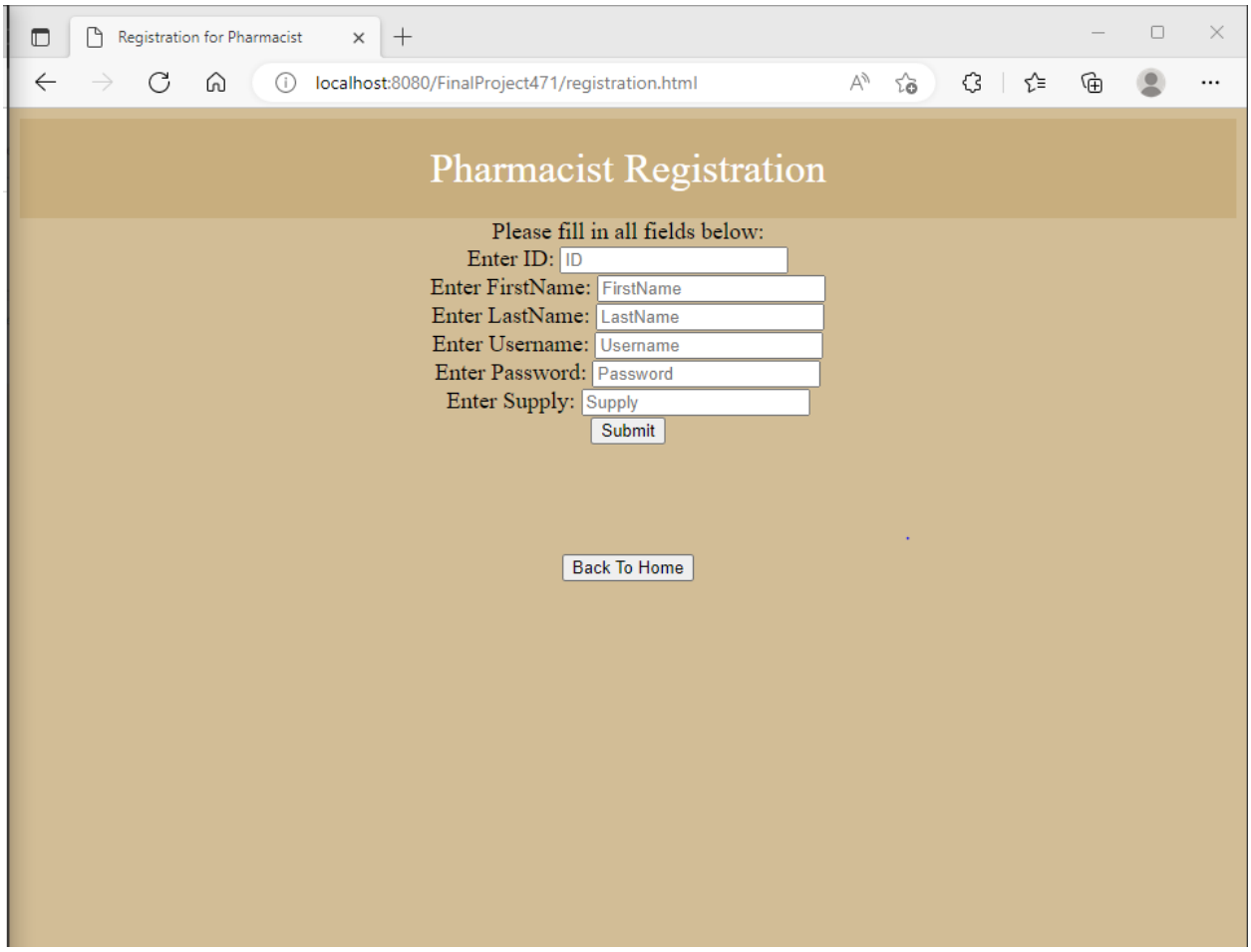


Back to the System homepage after logging out from doctor

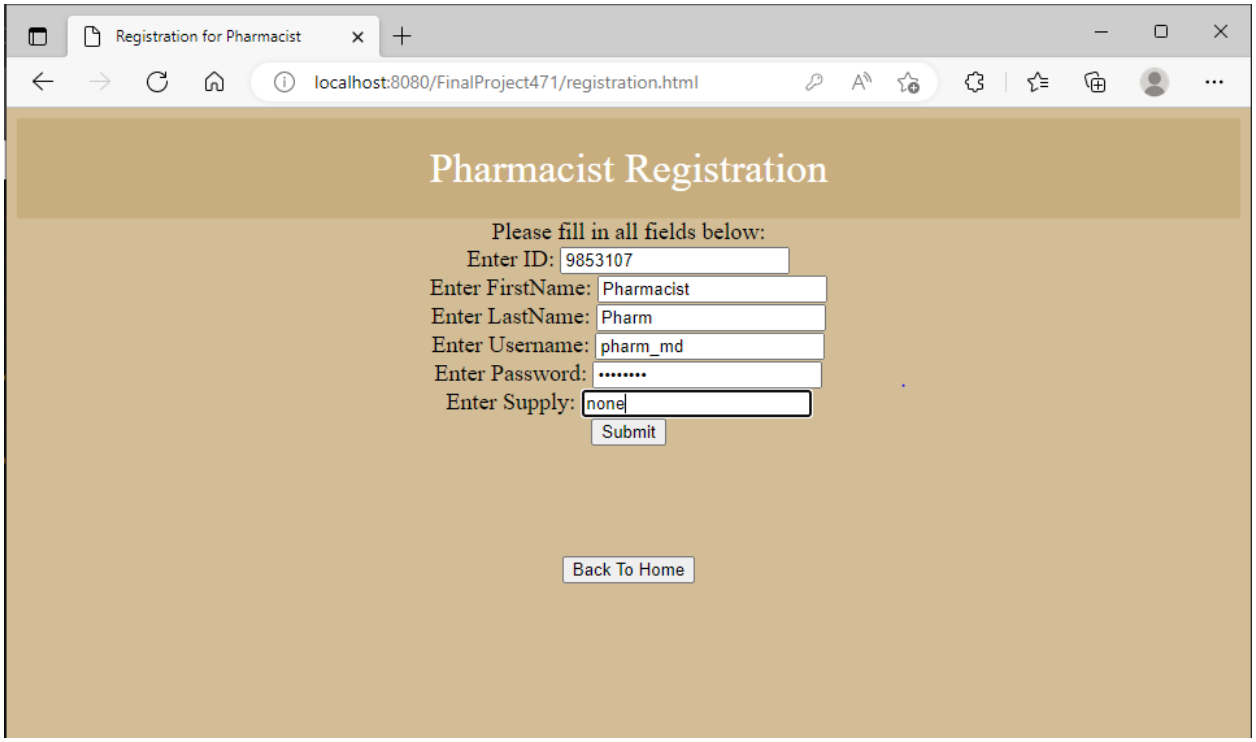
Now we can register a new pharmacist to do our example with. To do so, we must first press the register button underneath the logo. As seen before in the doctor registration, we are met with the user type selection page, which allows us to select our role via a dropdown menu.

User registration with dropdown menu

Select pharmacist, and press submit. Now, we have the registration form for a pharmacist as seen below:



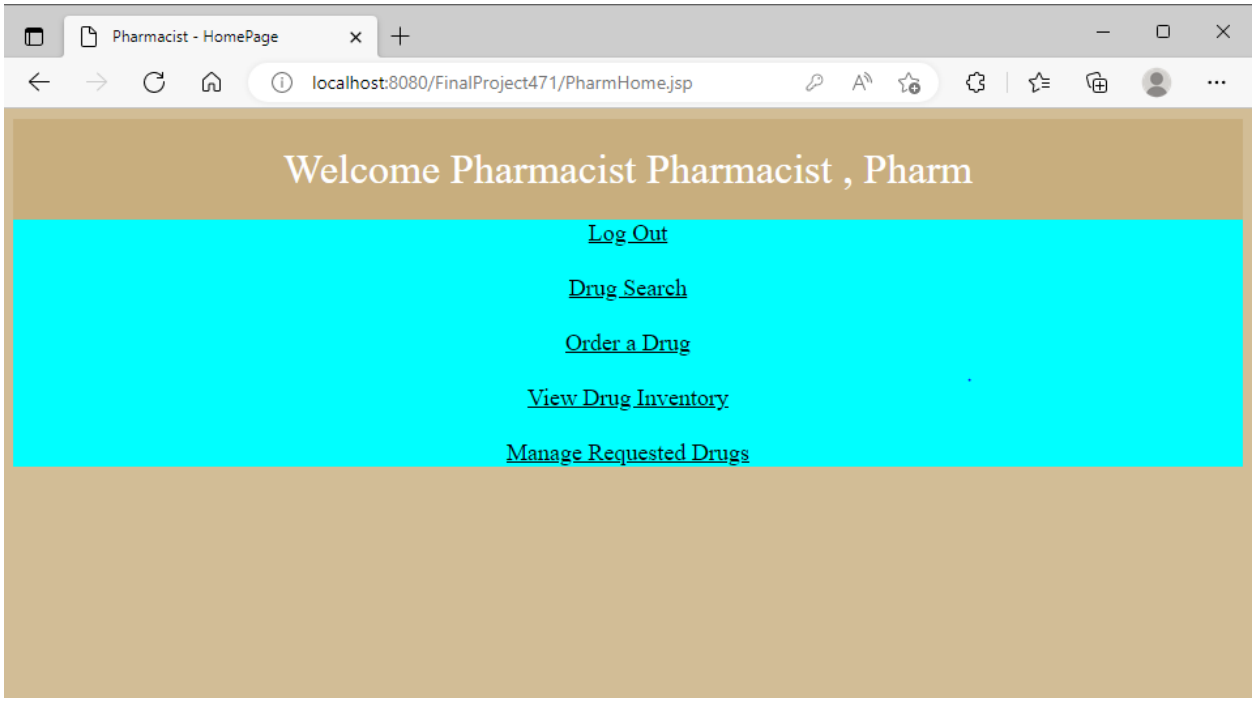
Pharmacist registration form



Sample fill-in form for our example

Now we can fill in the form as shown above to create a new pharmacist names “Pharmacist Pharm”. Please note the record of pharmacists in the database before registration in appendix u.1, and after this registration in u.11.

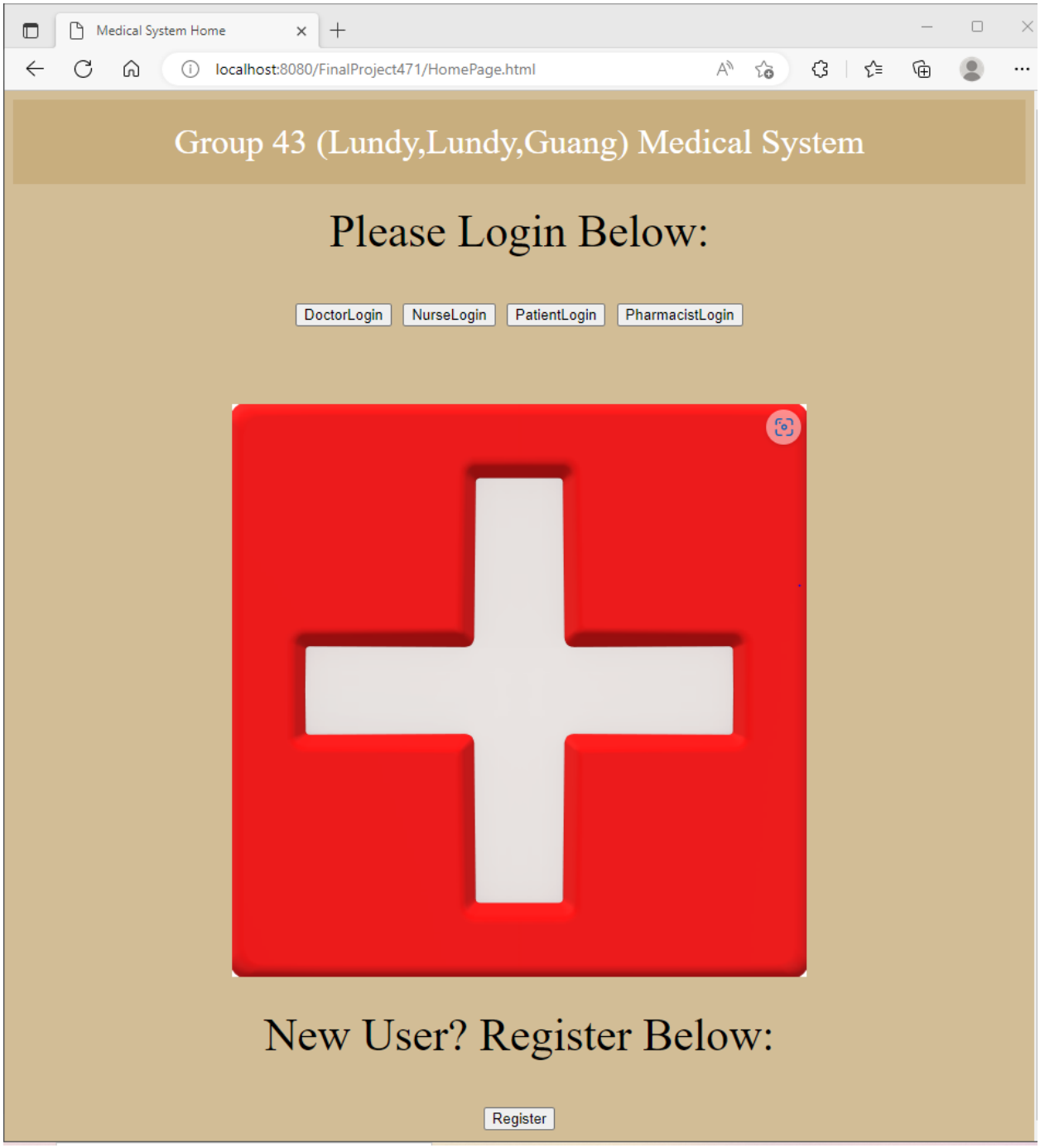
Upon submission, we get to our pharmacist homepage!



Pharmacist homepage upon registration

Now, lets go over the login/logout functionality of pharmacist. First, lets press the logout option seen above as the first option in the pharmacist's homepage.

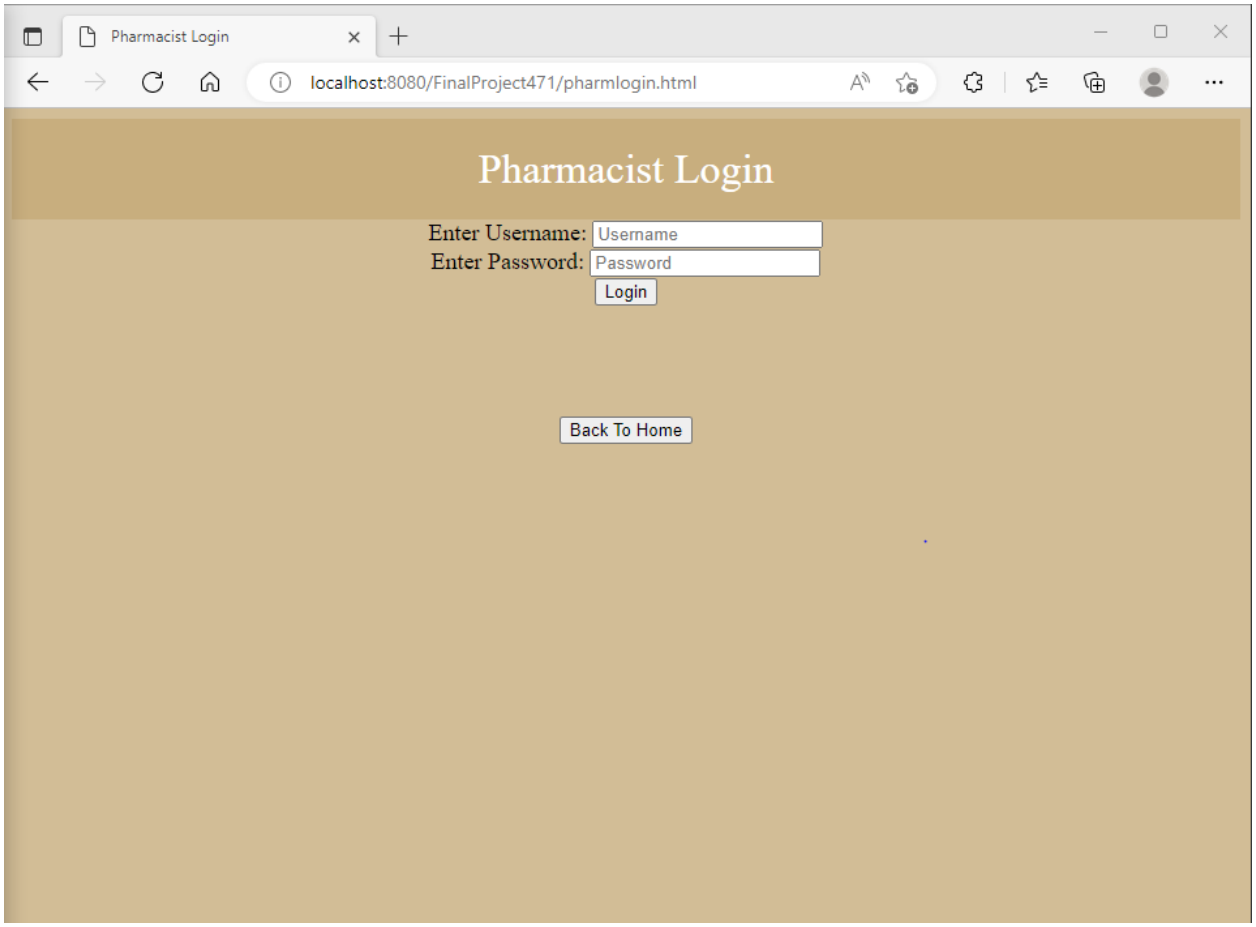
Doing so brings us back to the system homepage:



After logging out from our pharmacist user created

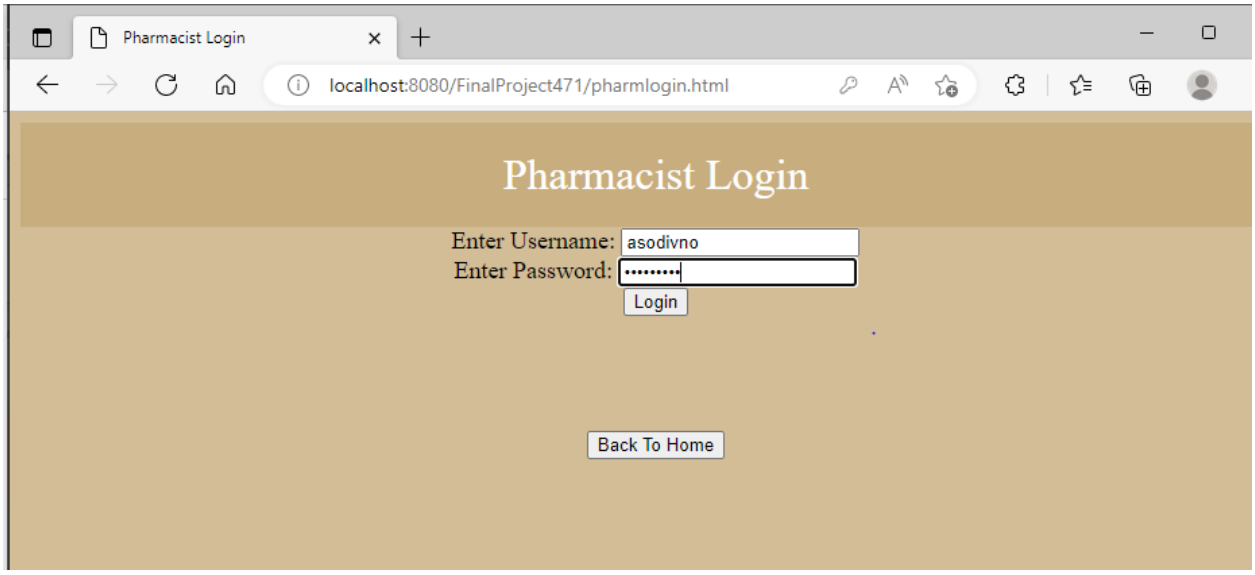
Now, we can attempt to login using the ‘PharmacistLogin’ button at the top left area of the homepage.

Pressing this button brings us to the pharmacist login page:



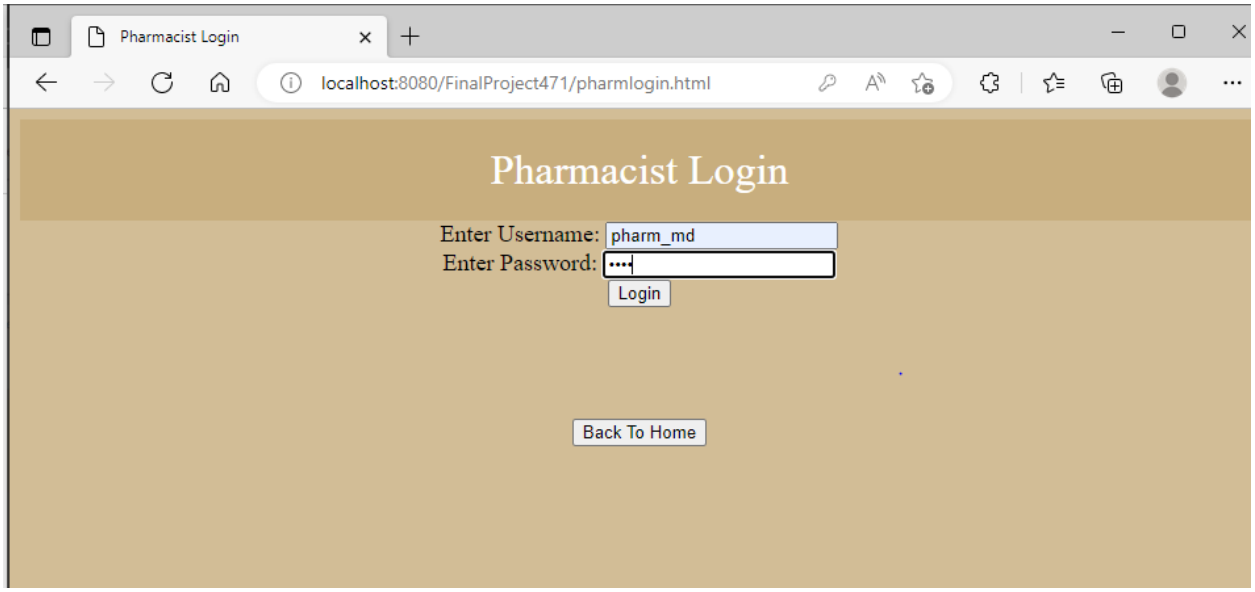
Pharmacist login

Now, lets try the (3) options available: try to login with a non-registered pharmacist, try to login with the wrong password only, and login correctly. The three options and their result from our system are shown in the following 6 pictures:

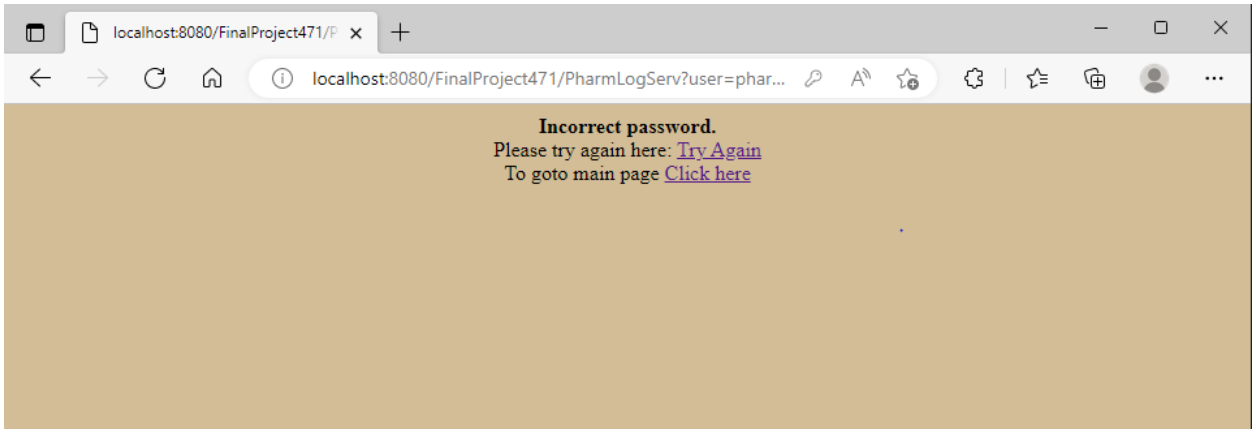


1a. Logging in with non-existing pharmacist

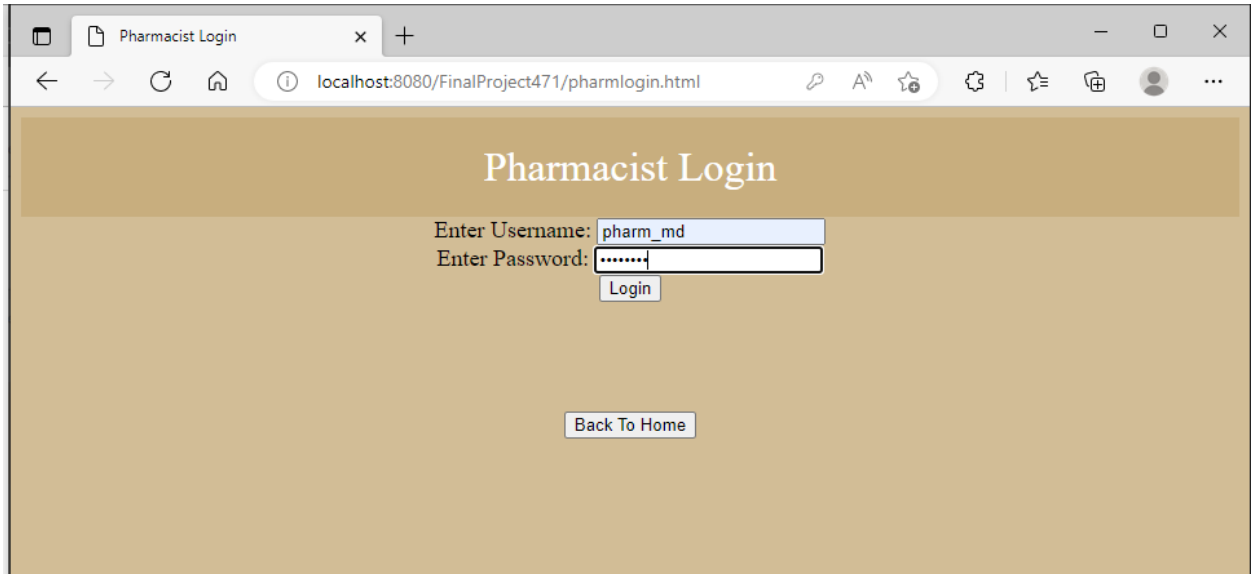
1b. Result of 1a. Note the link option to register or to go back to the system main page.



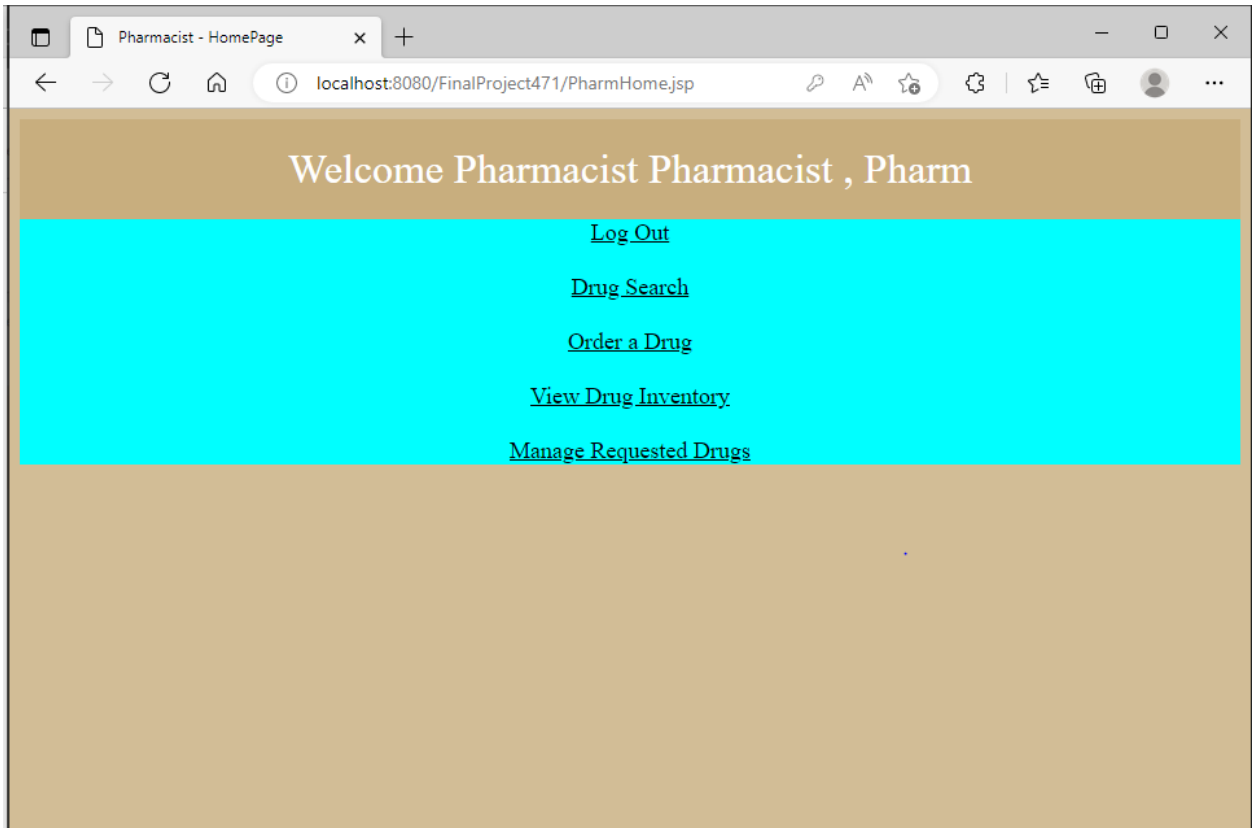
2a. Login with existing pharmacist, but with incorrect password



2b. Result of 2a. Note the option to try again (redirect back to login form), or back to the system homepage



3a. Logging in with the correct username and password

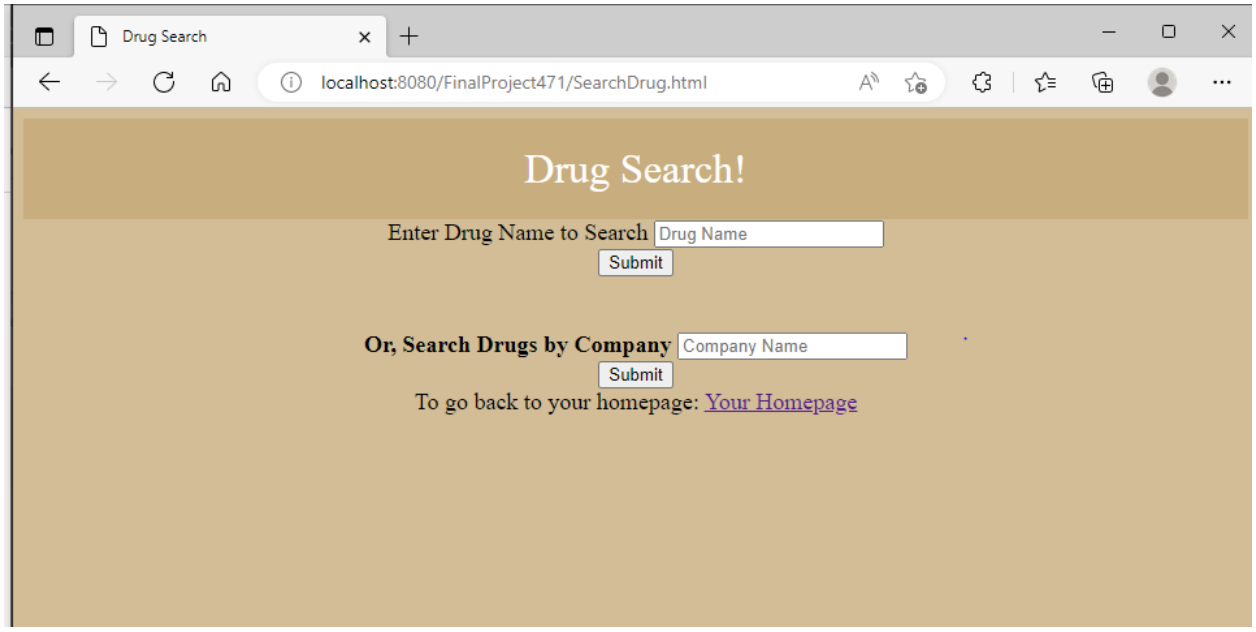


3b. Result of 3a. This brings us to the homepage for this logged in pharmacist.

Now we are logged in as a pharmacist named Pharmacist Pharm. Let us now go through each and every pharmacist option seen on the pharmacist's homepage and in the picture above.

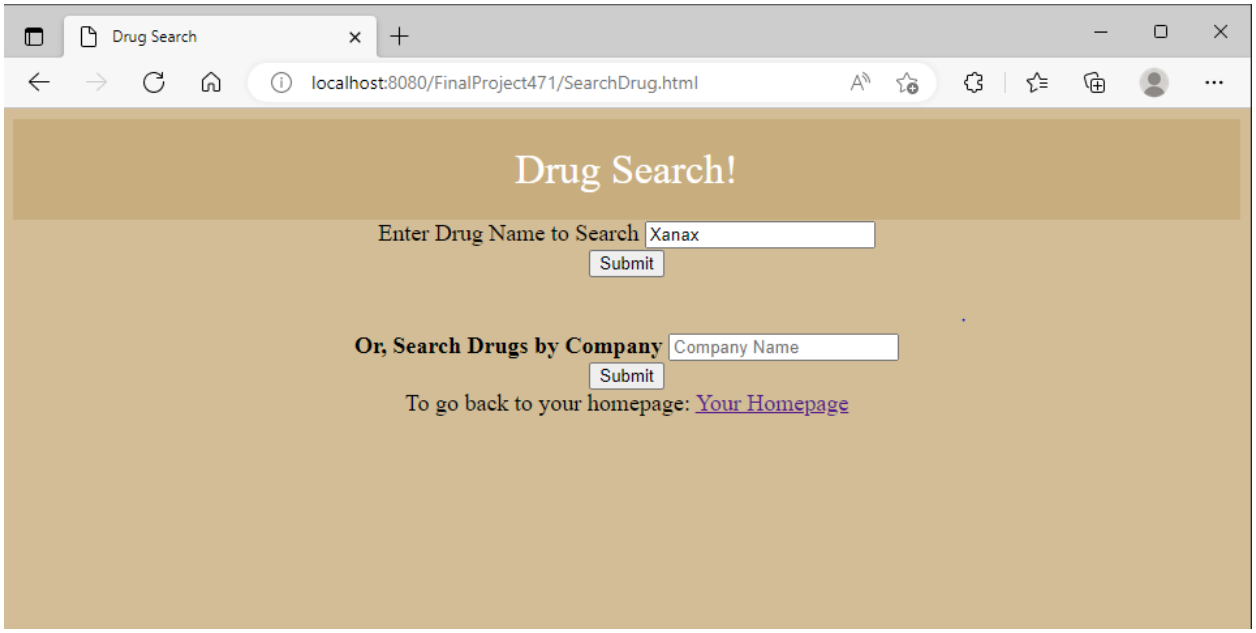
For ease, we will go through the options in order. Thus, we will begin our discussion with the Drug Search option.

Pressing drug search gives us



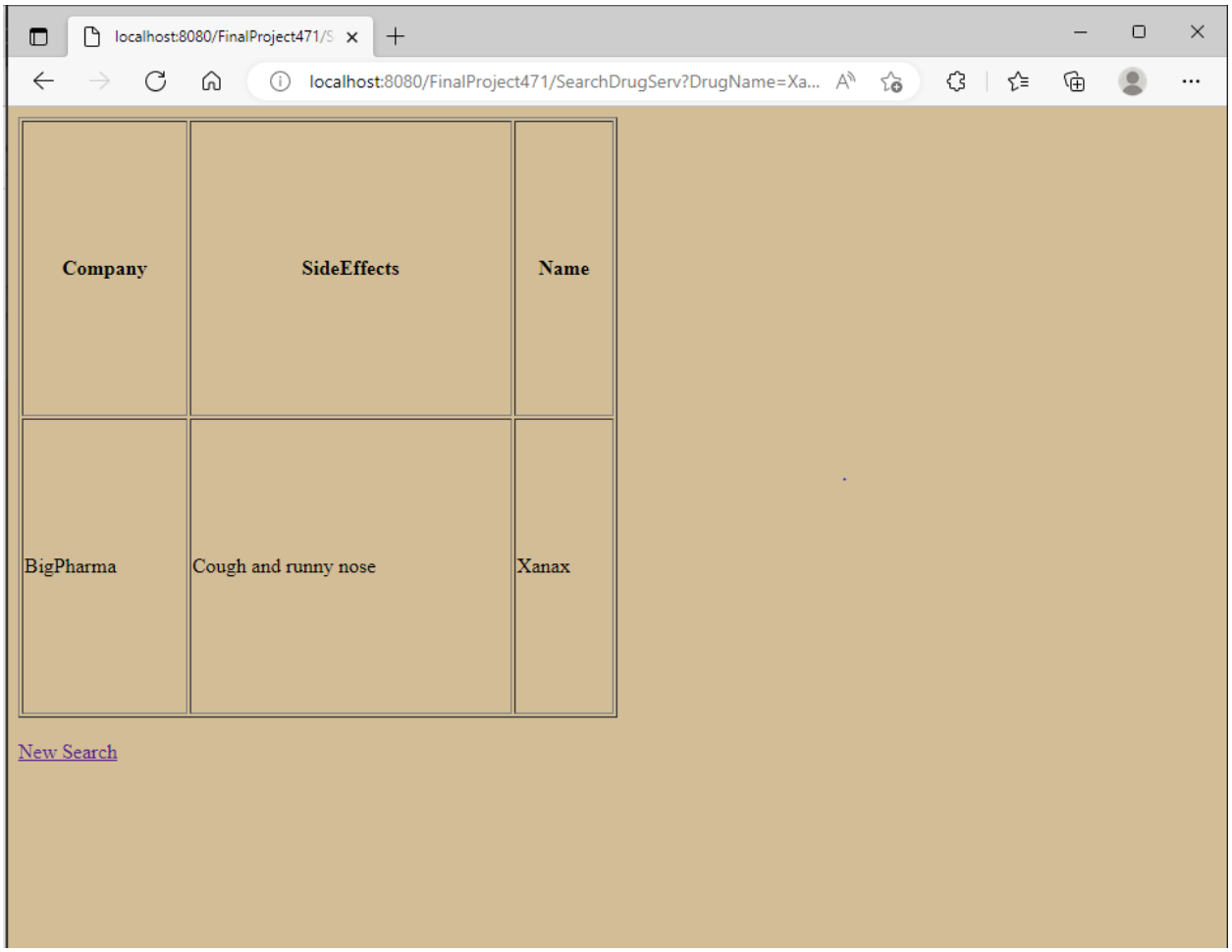
Drug Search page

Here, we have two options: search for a drug by name, or search for a drug by its company name.
Lets first search for the drug Xanax by name to see the search by name function:



Sample search by name for drug xanax

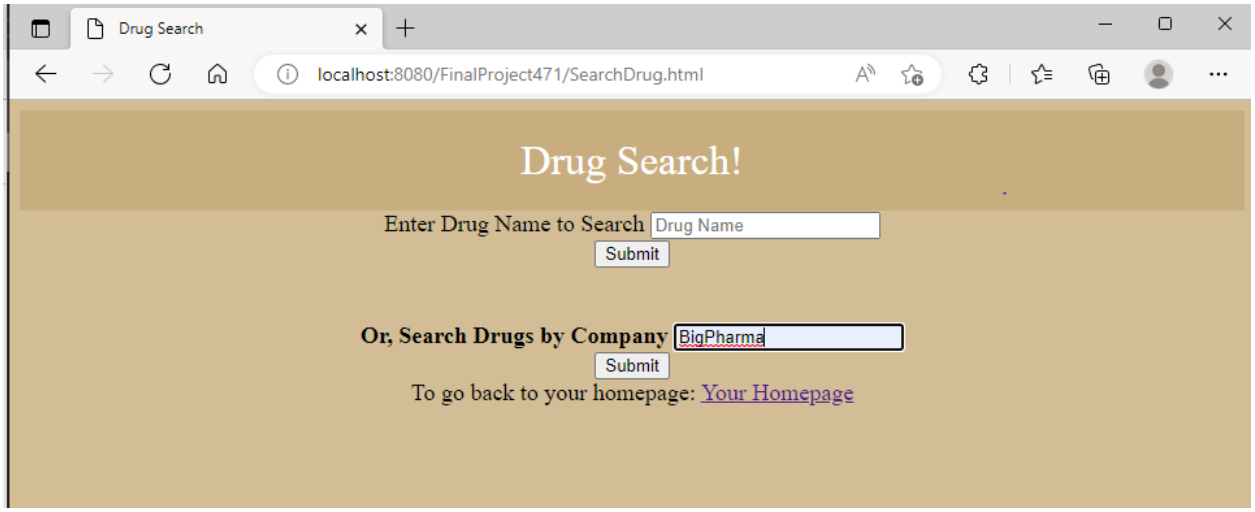
Upon hitting submit, we get the following result, which effectively matches the database for that search and provides all details regarding that drug that is stored in our system.



Result of search for name 'Xanax'

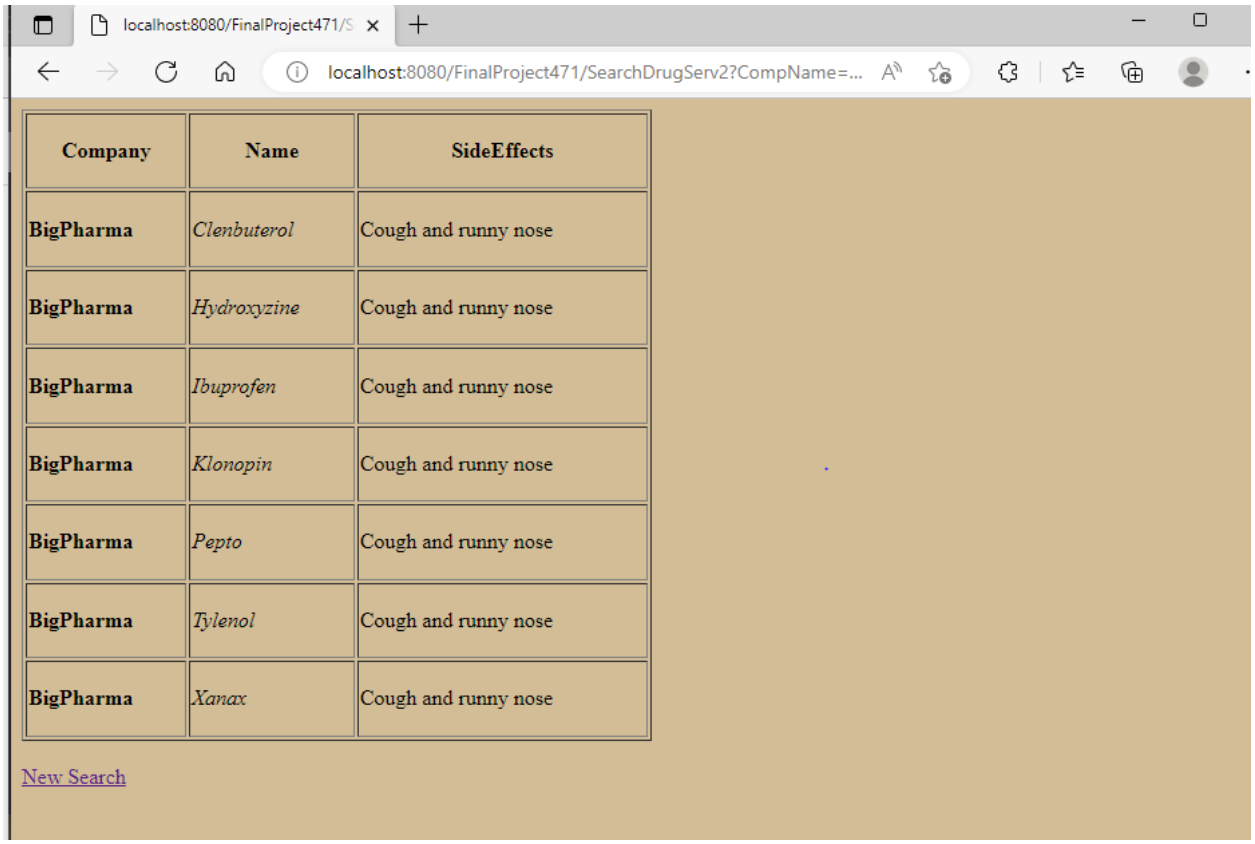
Now, lets try another search by hitting the link below the table 'New Search'. This brings us back to the drug search.

Now lets try searching for drugs by the company BigPharma to see information on all drugs provided by this company.



Searching drugs by company 'BigPharma'

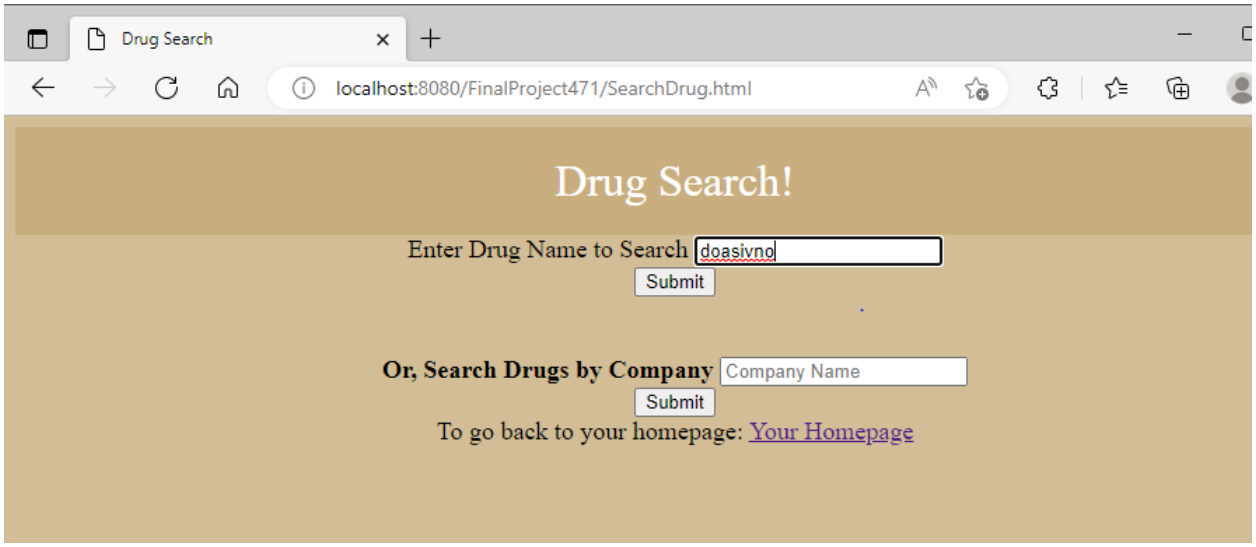
The result is shown below. The system returns all information on all drugs that are provided by that searched company rather than a single drug.



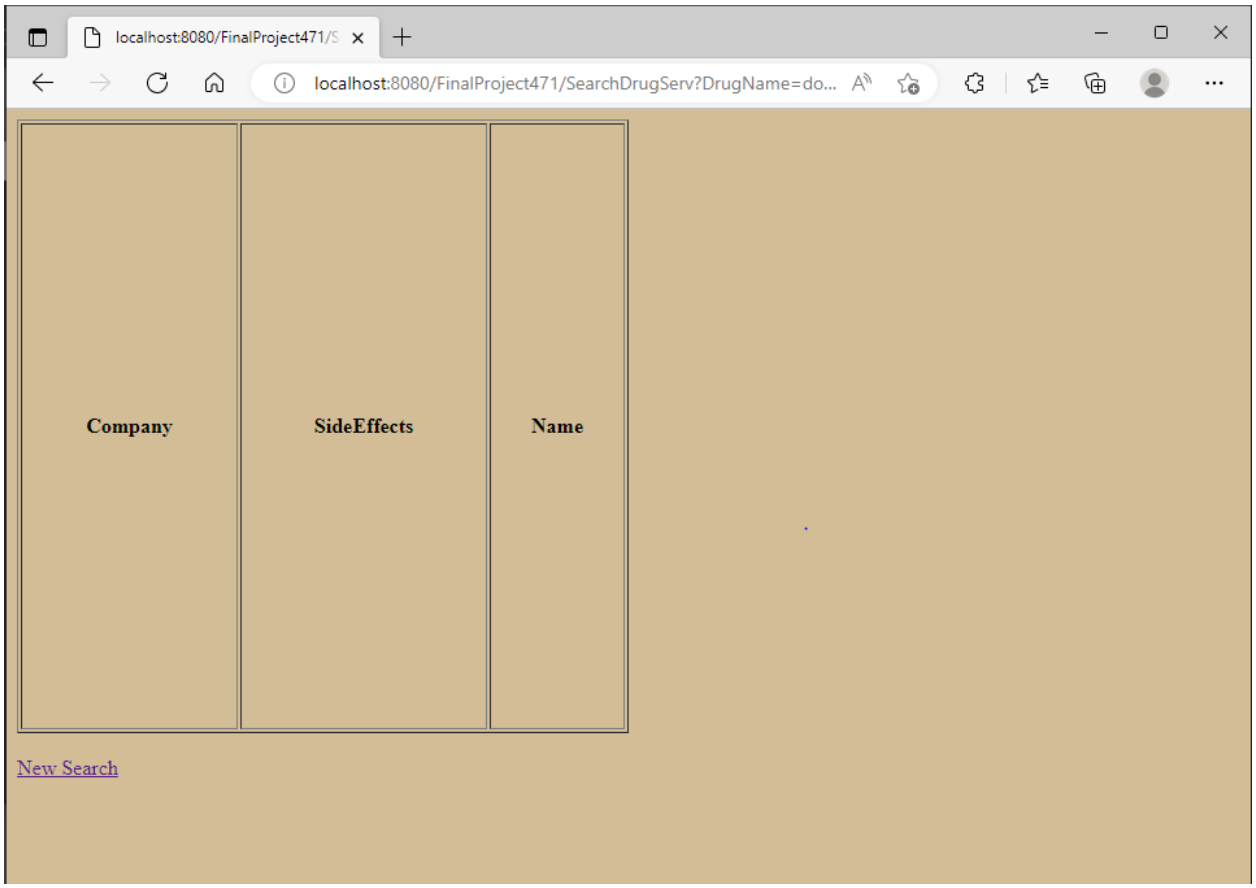
Search result for company BigPharma

Now we will go back and see briefly these two searches with non-existent drug names and non-existent companies. Both options print out empty tables, an expected result. The two

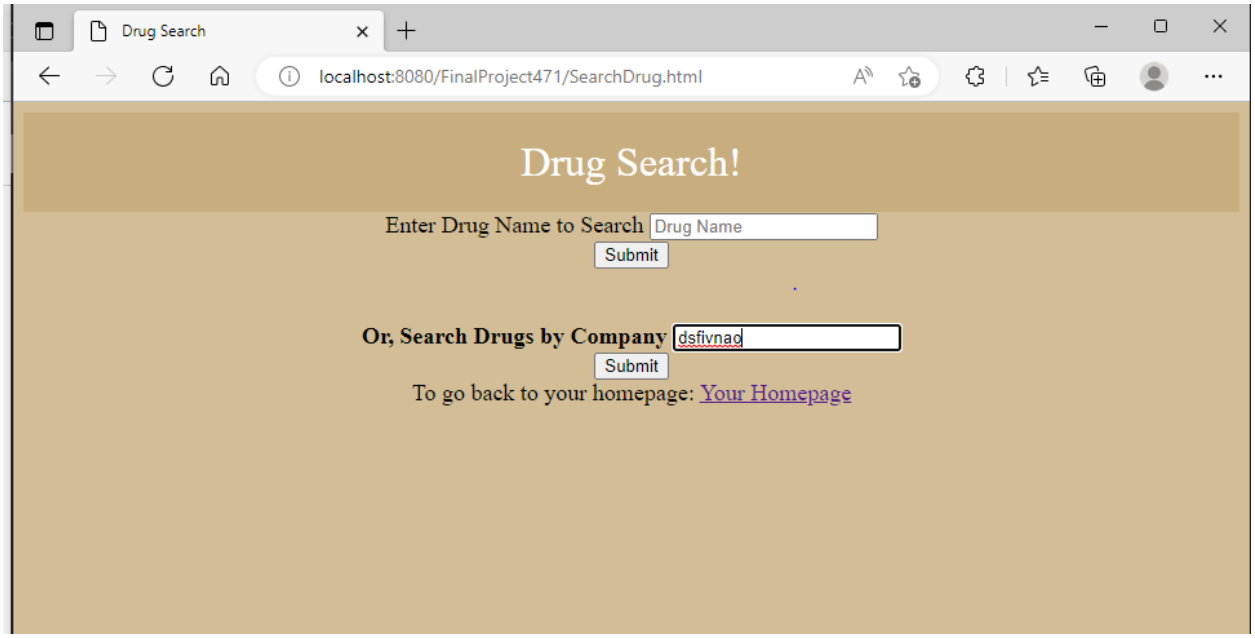
searches and the two corresponding results are shown in the snapshots below:



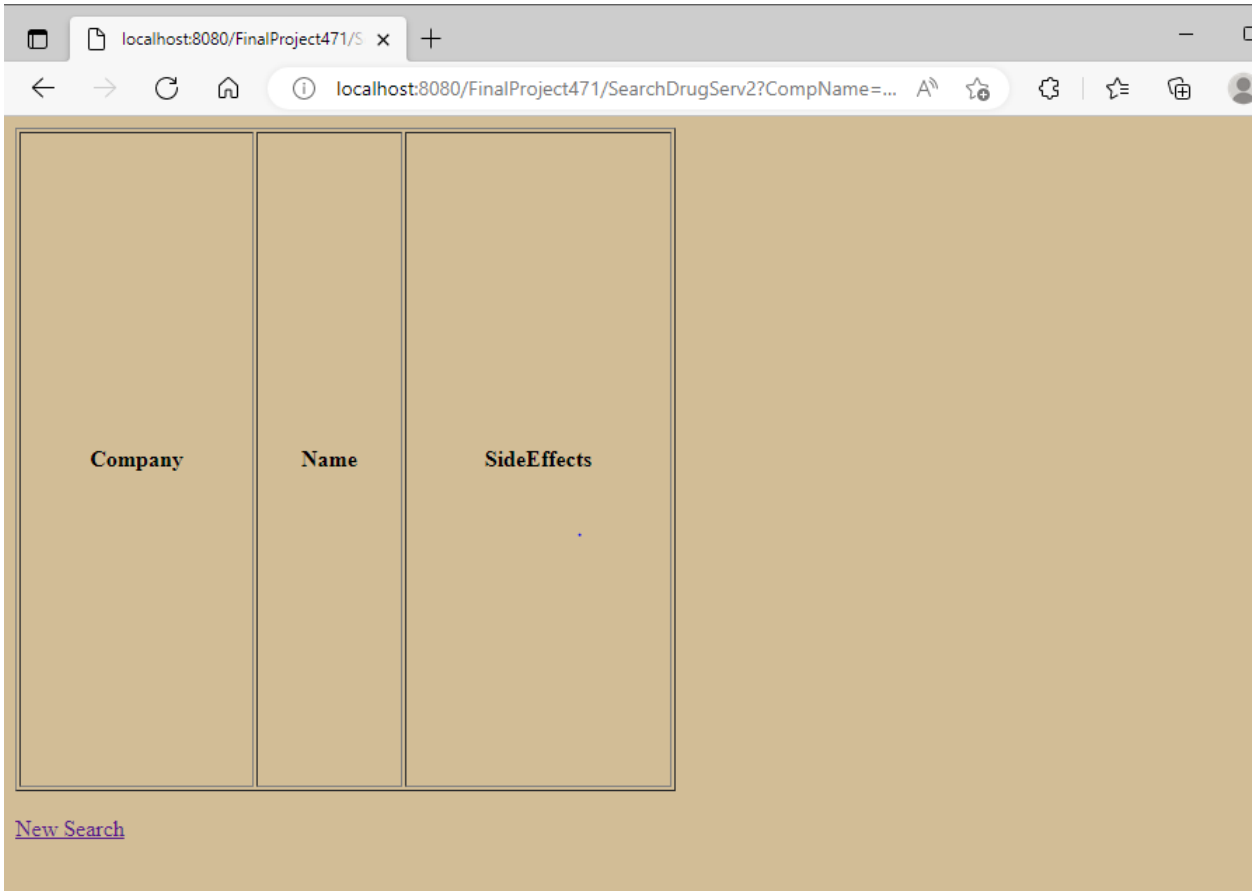
Non-existent drug name search



The result of above

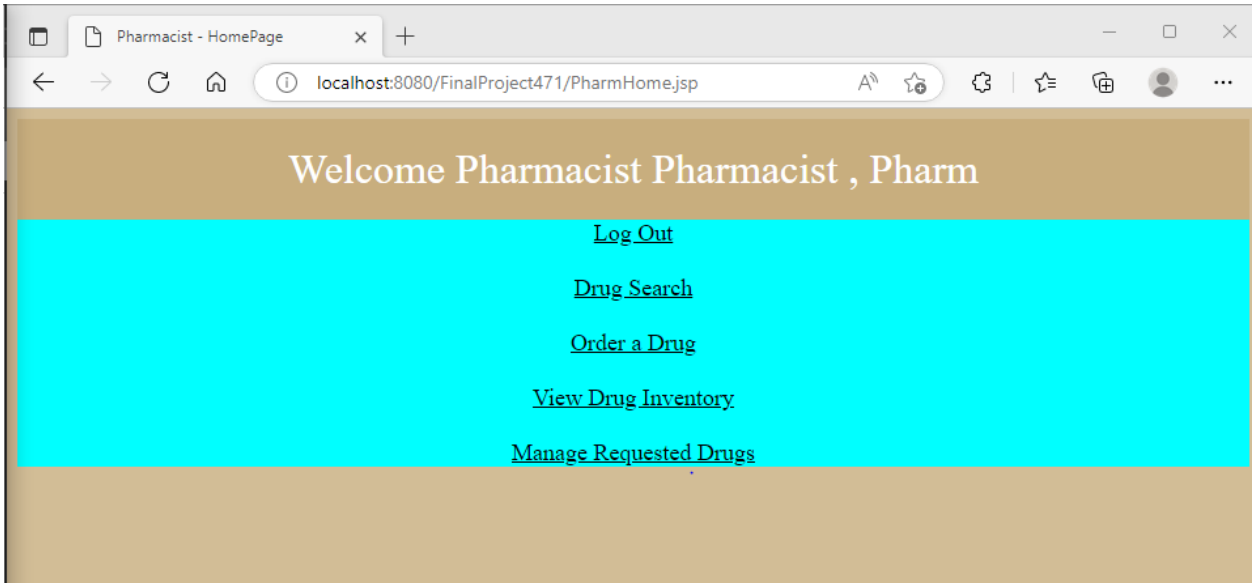


Searching a non-existent drug company



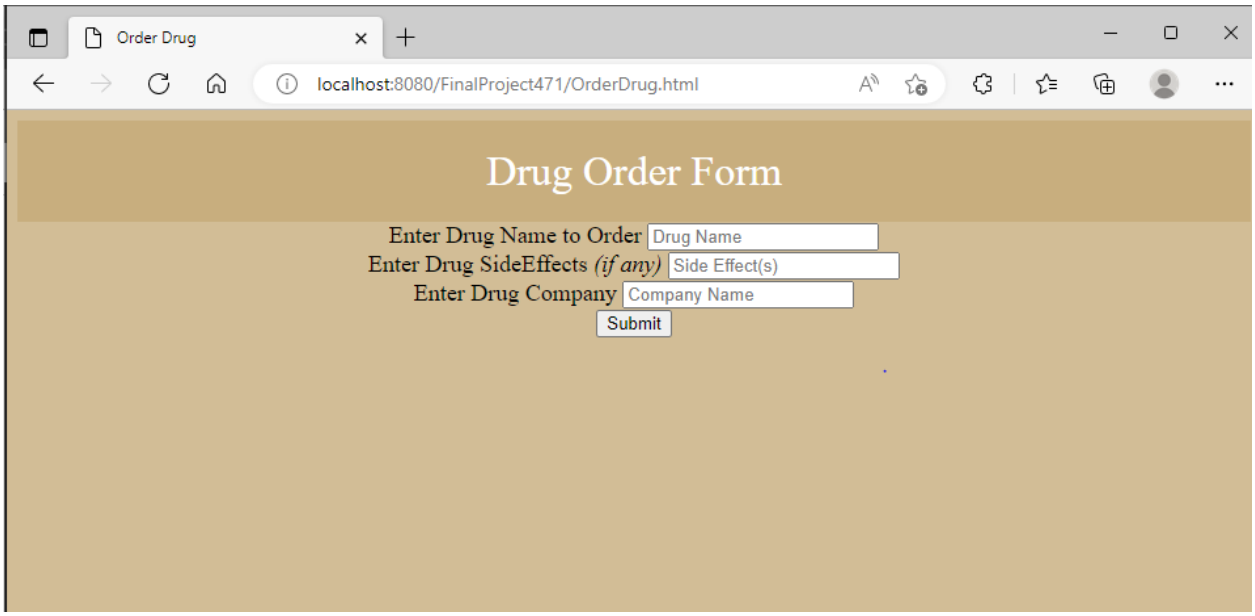
The result of the above search

Now we have seen the drug search functionality for a pharmacist. Lets go back to the pharmacist homepage (New Search -> Your Homepage). Now the next option to choose is the Order a Drug option.



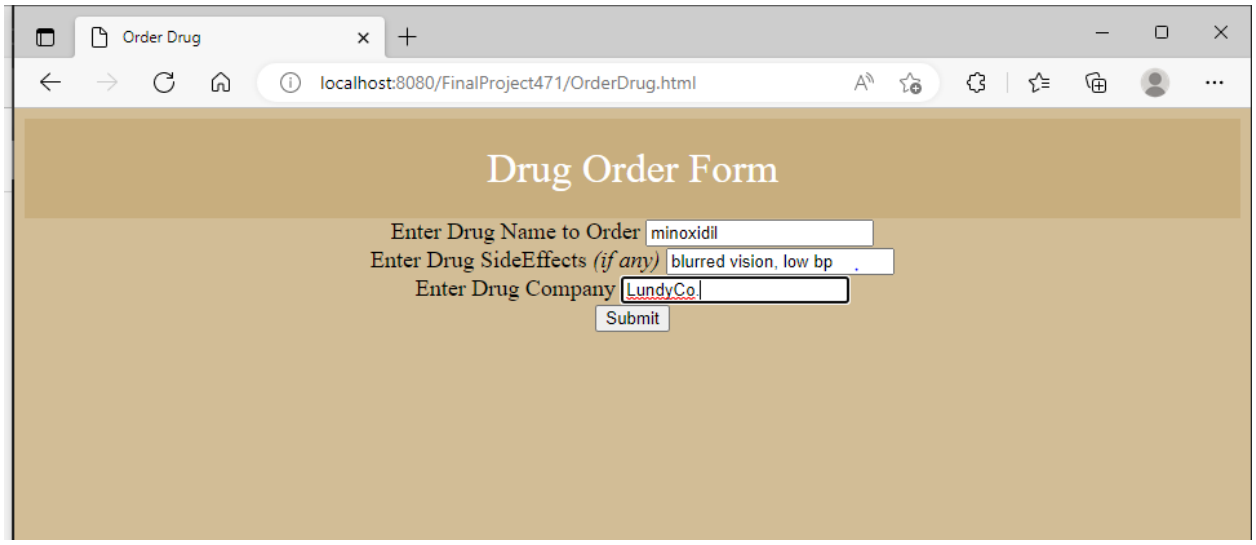
Again, the pharmacist options.

After selecting order a drug, we get the order a drug page/form:

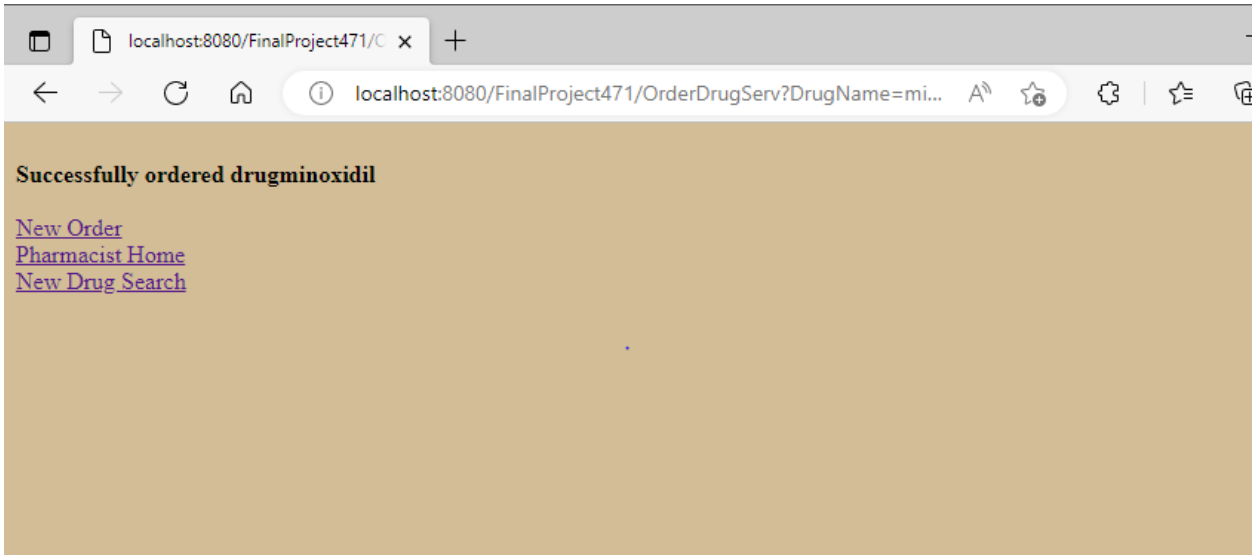


Drug Order form

Let us first discuss the option of ordering a unique drug. To do this, we will enter the form as such:



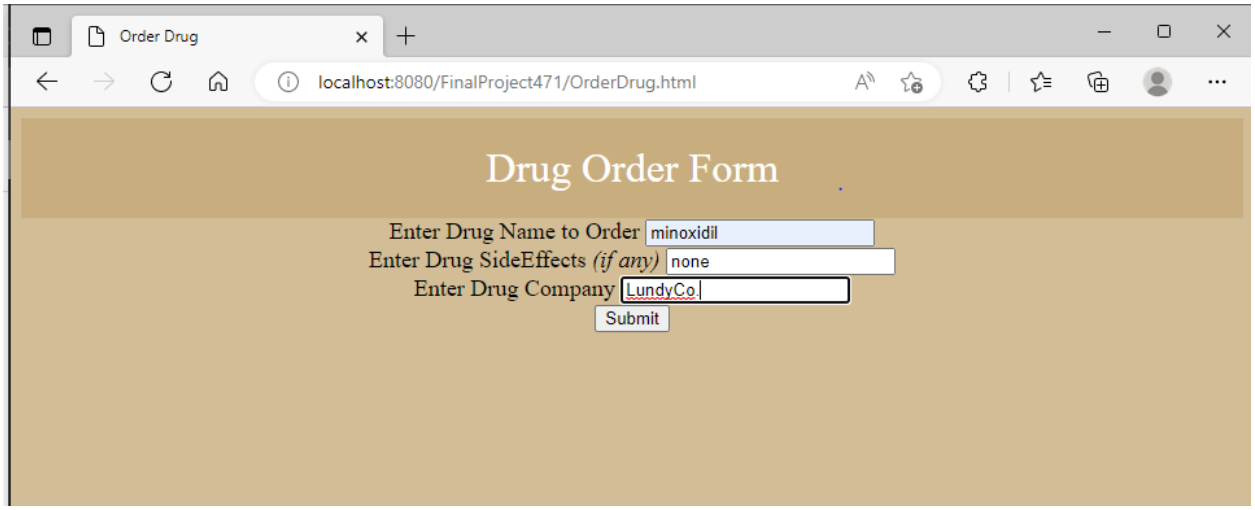
Ordering a new drug.



The result of the above order form submission.

We see that the drug was successfully ordered (refer to appendix u.12 to see the database change). Now we have the option to search a drug (as previously discussed), pharmacist home (back to the pharmacist homepage), or new order to bring us back to the drug order form to try another one.

Lets show the other option: trying to order a duplicate drug. Lets select ‘New Order’ and try to order minoxidil again to see how our system reacts to duplicate order requests.

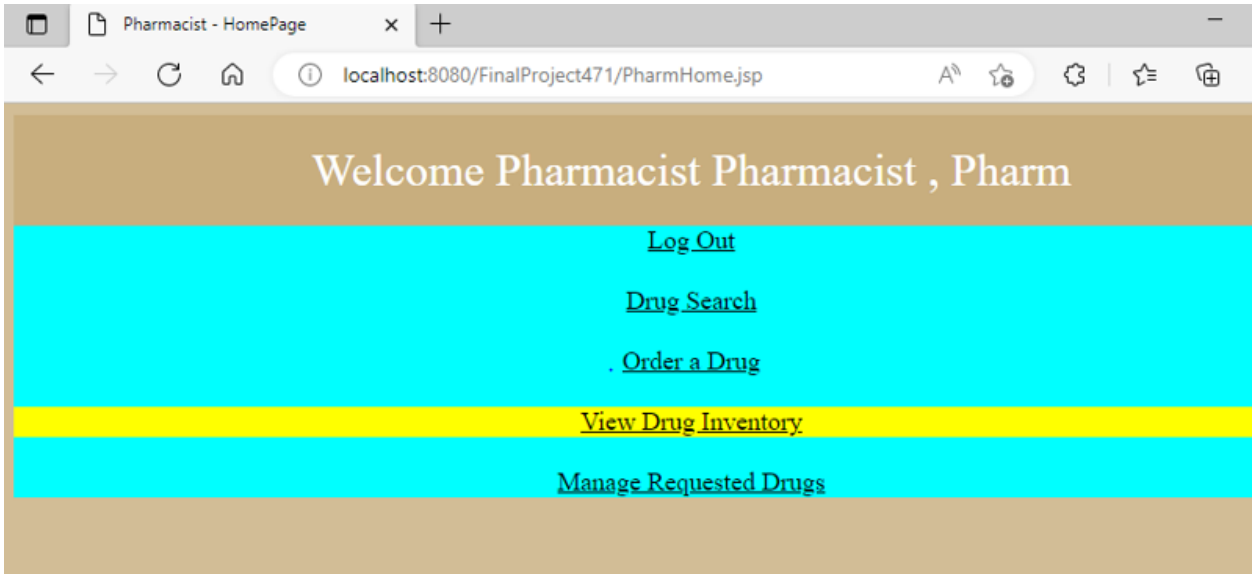


Attempt at a duplicate drug order

The resulting page from the above order attempt

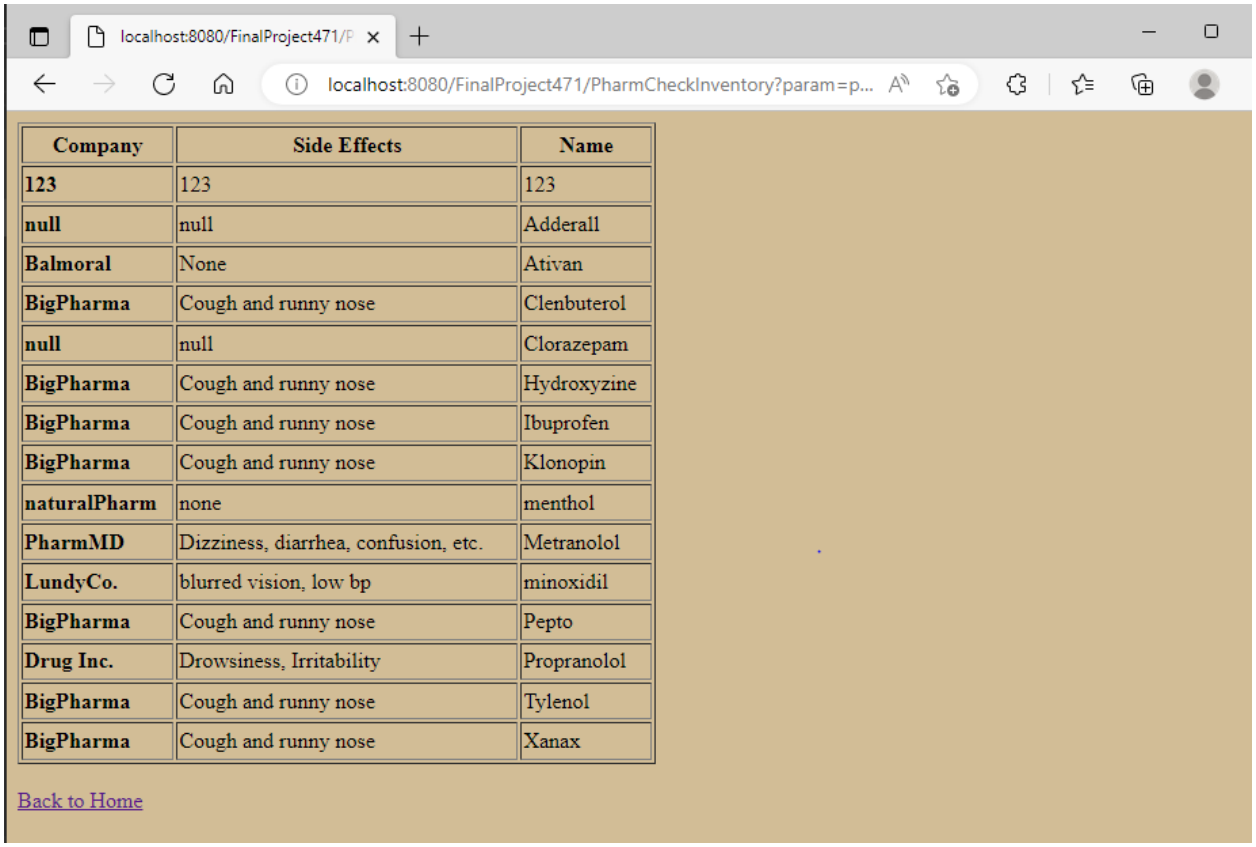
As seen above, the system does not allow a duplicate minoxidil drug to be ordered, and can be seen in appendix u.13.

Now we can select pharmacist home and move on to the next option on their homepage: view drug inventory.



Selecting view drug inventory from pharmacist homepage

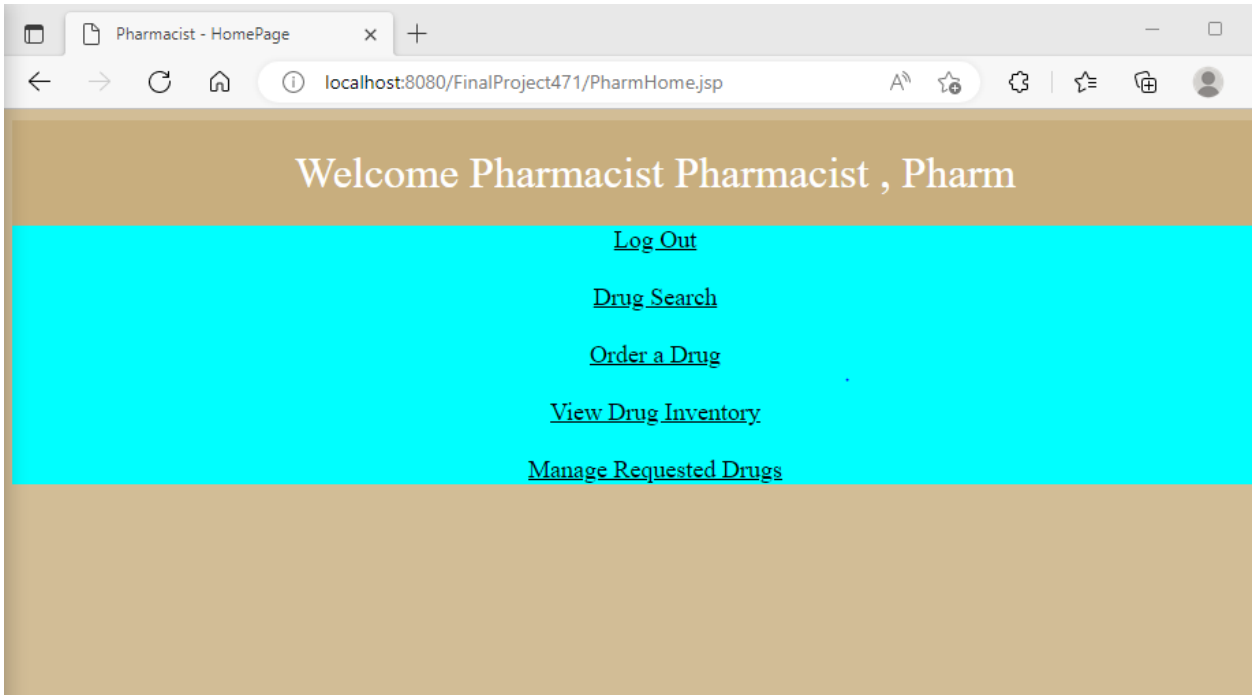
Selecting this retrieves the entire system drug inventory for a pharmacist to view:



Retrieved pharmacist drug inventory once view inventory is selected.

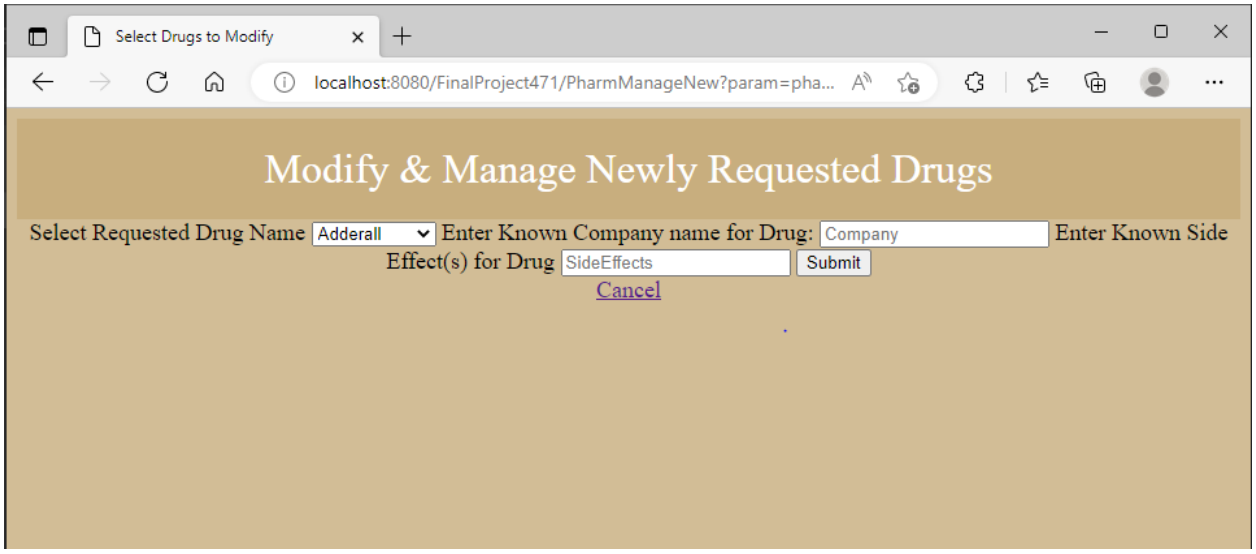
Notice the NULLs in the drug inventory. As previously discussed, all text forms of our website are protected from NULLs and empty inputs. Therefore, all NULLs present here are from the

doctor request we mentioned before, where they request a drug and a pharmacist must later fill in that drug's information for completeness (but doctors may still prescribe the drug before pharmacist completion).
Speaking of that, we can now go back to the pharmacist homepage and examine the final pharmacist option: manage requested drugs. This directly ties into what we just discussed. This manage requested drugs gives the pharmacist the option to monitor the database for any newly requested drugs from doctors (that will have NULLs for information attributes), and can then fill in the information for that drug to complete its integration into our system. Thus, it completes the doctor drug request previously discussed to obtain a completely integrated drug into our system, rather than a requested drug with NULL values.
Now lets show this option by pressing manage requested drugs from the pharmacist homepage:



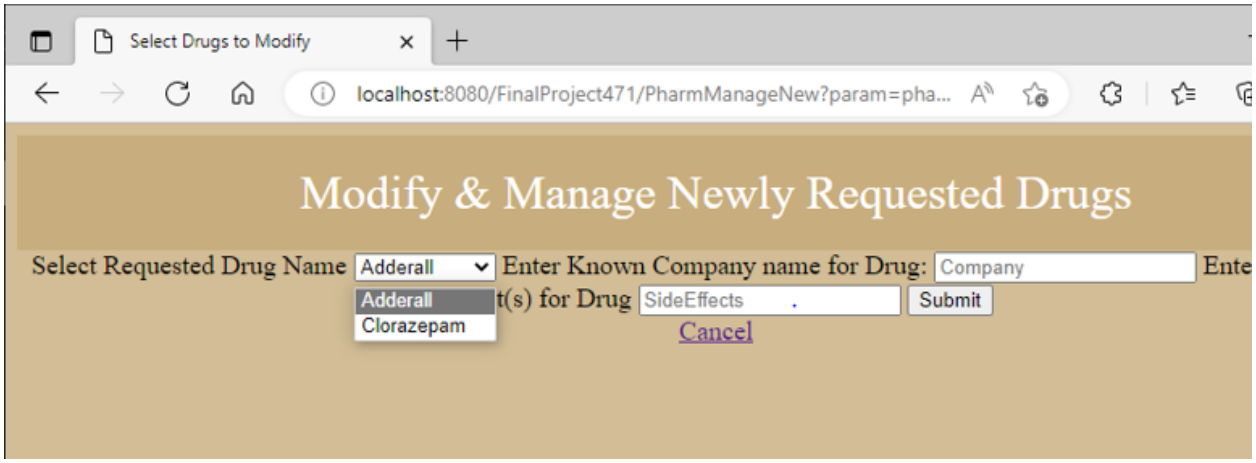
Pharmacist homepage which includes the final option: Manage Requested Drugs

Selecting this gives us this page:



Manage drugs page

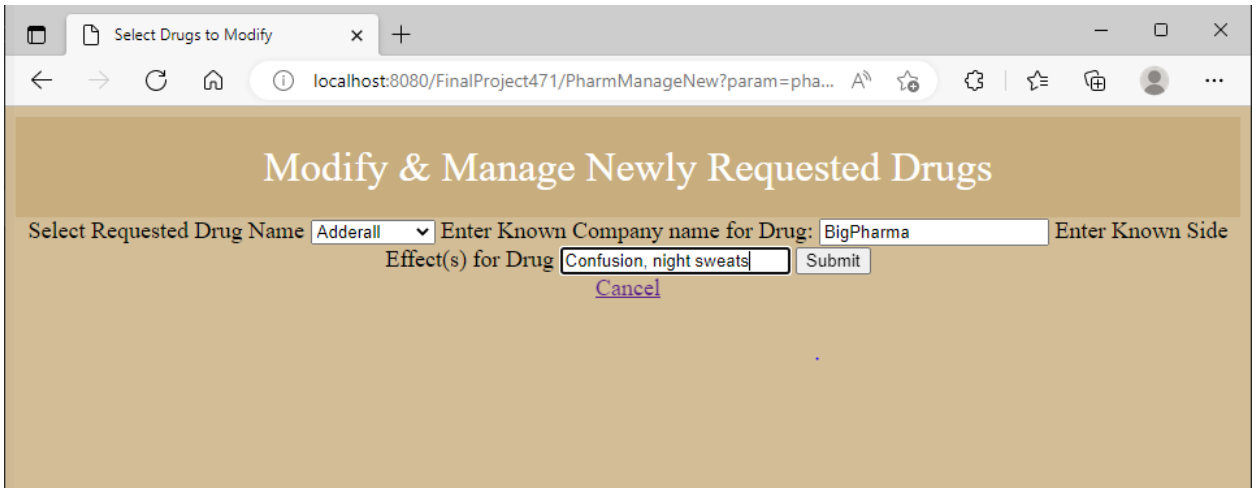
Here, we are presented with the page to fully integrate and manage drugs that doctors have newly requested since the pharmacist's last login. As you can see, we have a dropdown menu (which has every newly requested drug which has NULL attributes as an option), as well as the text field to fill in the drug info for that drug.
Thus, we can select a newly requested drug and fill in the Company Name and the Side Effects for that drug to complete the request from doctor to pharmacist.
Taking a look at appendix u.14, we see all the drugs that have been newly requested, including the adderall request we made when discussing the doctor-side of requesting a new drug. We see that these drugs exactly match the options for the dropdown menu (seen below):



Dropdown menu to show all options for newly requested drugs

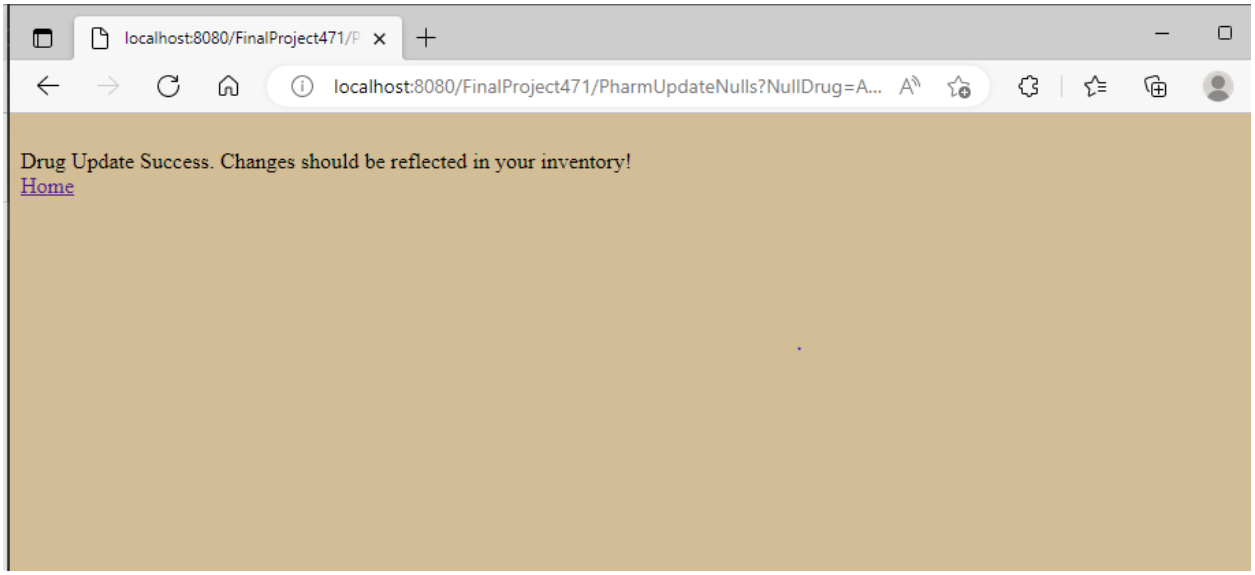
Note that these only include drugs that have been recently requested and no pharmacist has filled yet. As soon as a pharmacist does this, it will no longer be null and thus no longer available in the dropdown menu, since the drug will now be integrated and no pharmacist needs to take any action for that drug.

Now we can fulfill the adderall request we created as a doctor earlier. The form entry is shown below:



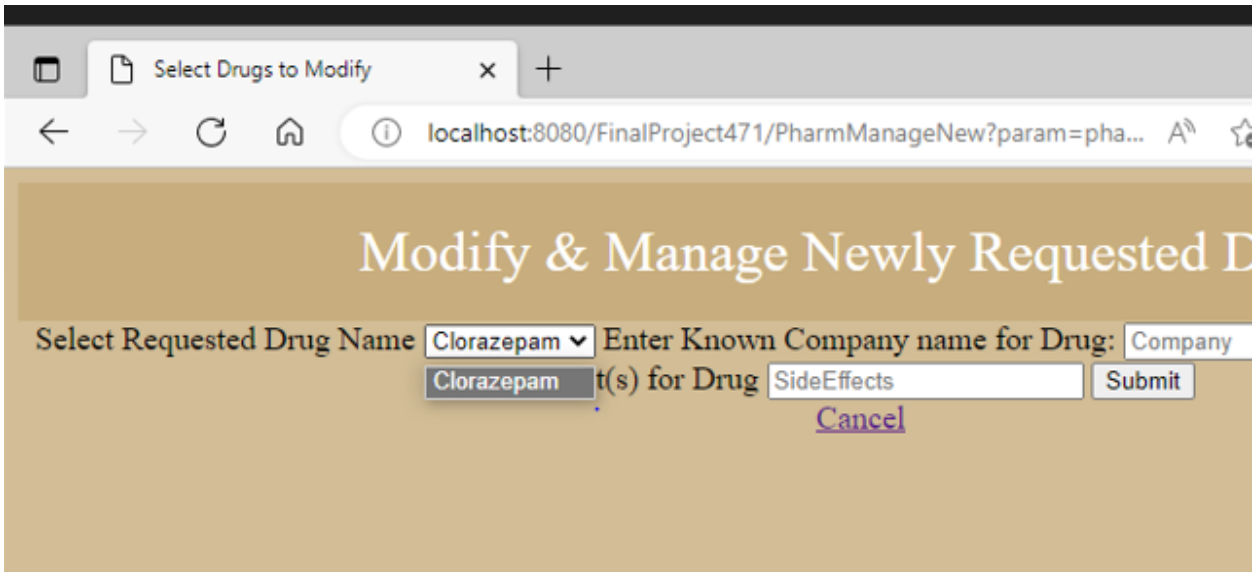
Form submission sample to fulfill the adderall drug request made previously from a doctor

Pressing submit will update that drug, while pressing cancel will bring us back to the pharmacist homepage, and therefore that drug will still need to be fulfilled by a pharmacist (but can still be prescribed by doctors).



Submitting the above form - request is successful

Please view appendix U.15 to see the database change for the previously NULL adderall. Now, patients for example can search the side effects or the pharmacist can see that drug in a company search since both those fields are no longer NULL. Now, lets go back home and see that adderall is no longer part of the dropdown menu, since no action is urgently required from the pharmacist for that drug.

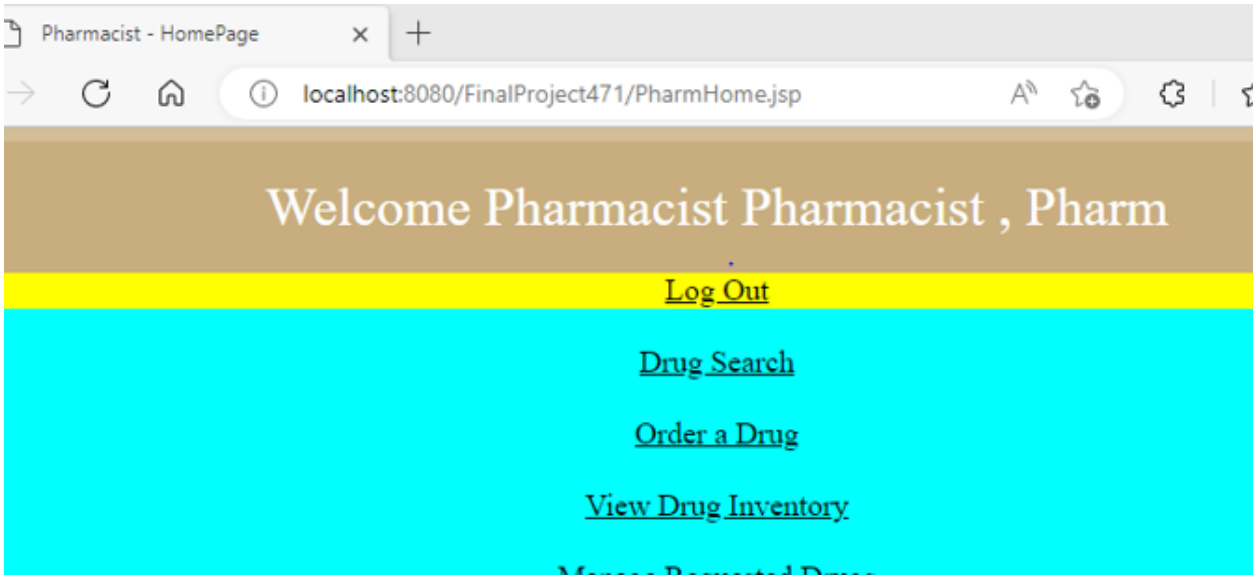


Going back to manage drugs, we see that Adderall is no longer an option to be managed since we successfully fulfilled that drug's request.

That concludes our discussion on pharmacists, now we can show how a nurse can use our system.

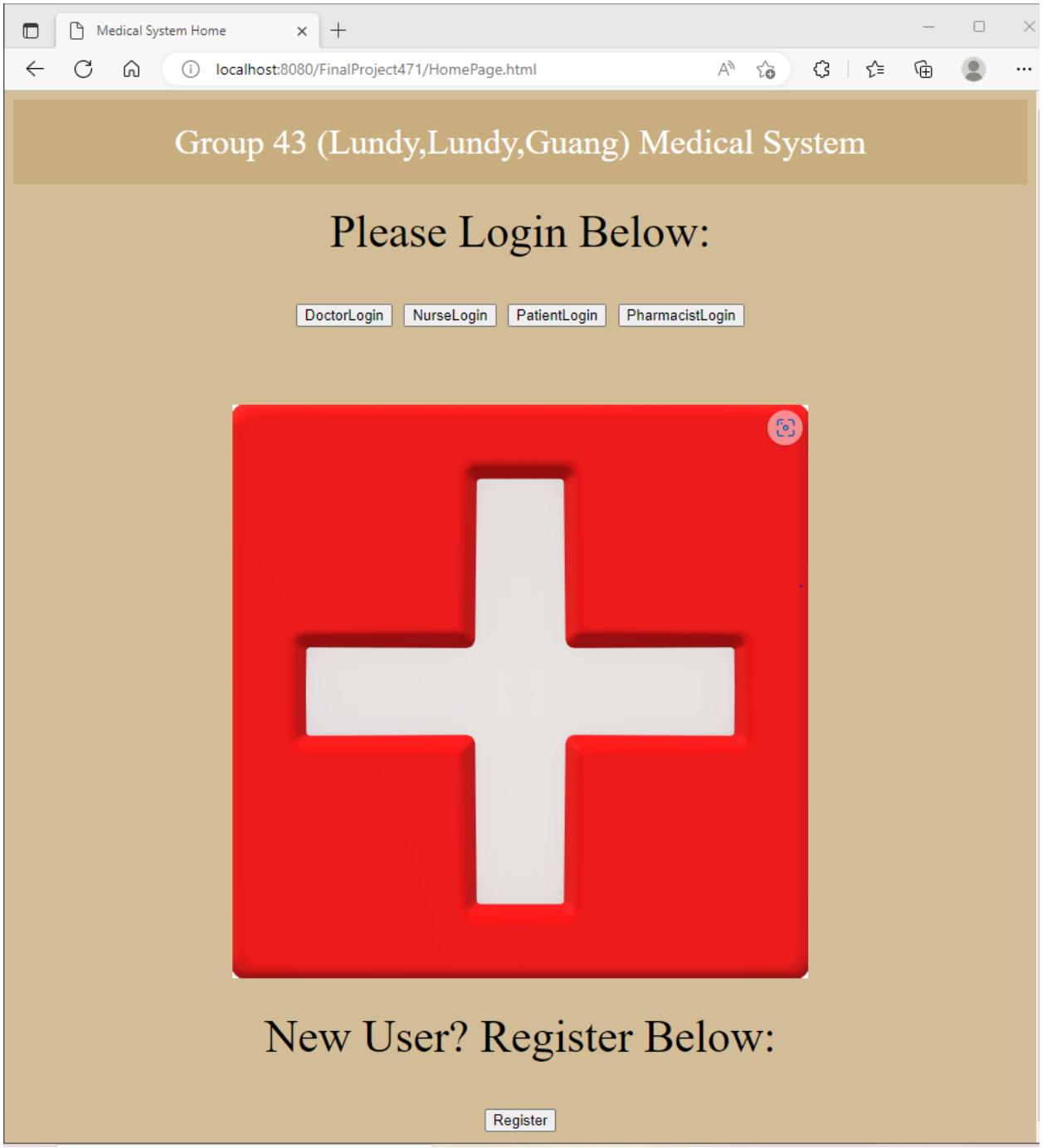
Nurse Functionalities

To begin our discussion on how a nurse can use our system, we first logout from pharmacist.



Pharmacist logout

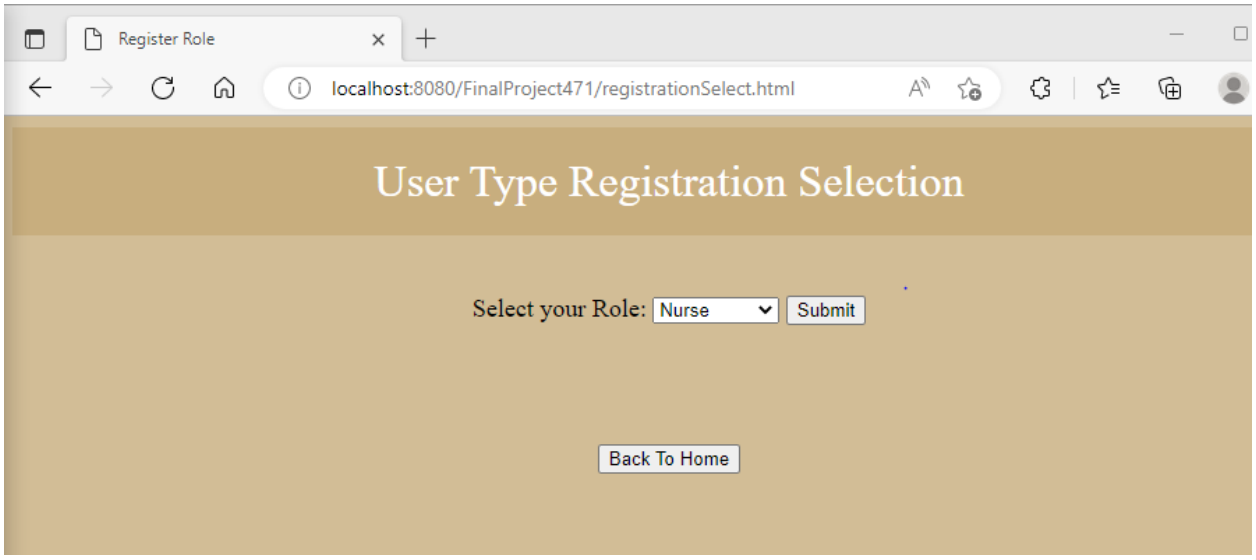
Now we are brought back to the familiar system homepage:



System homepage

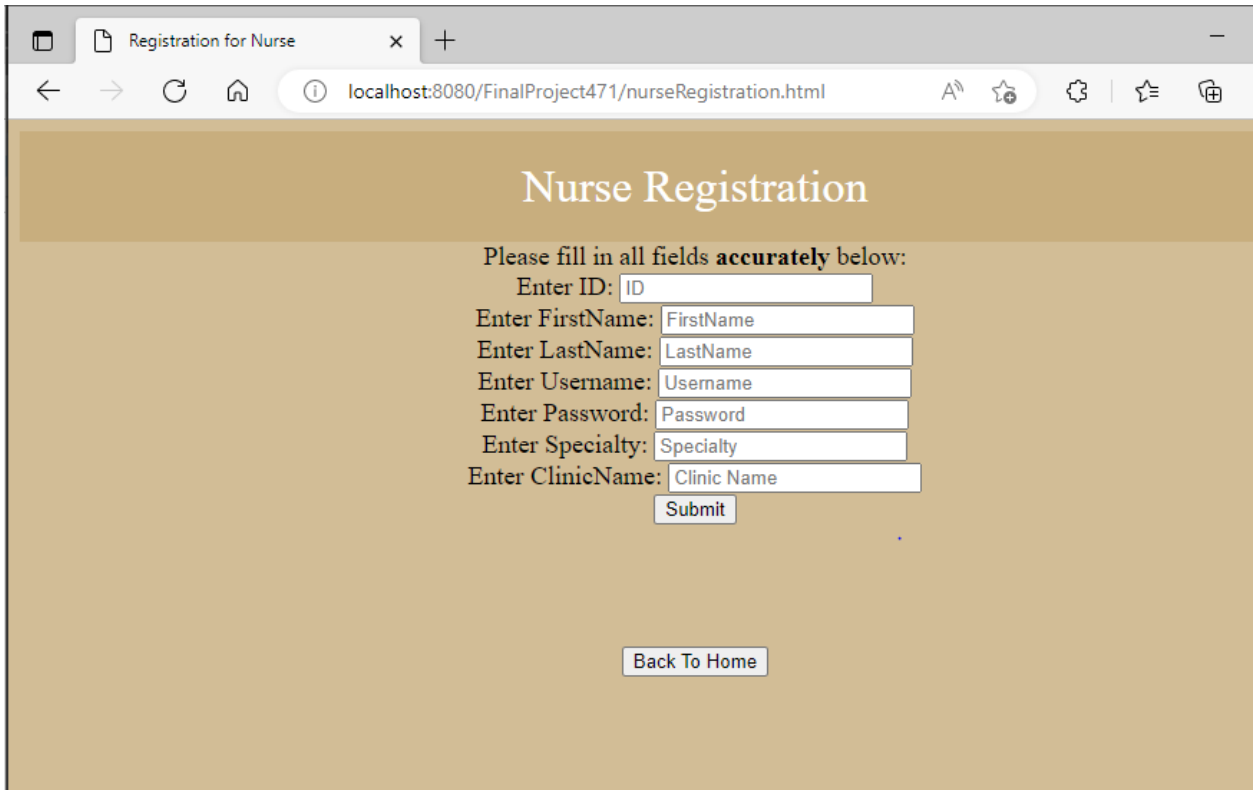
As we did with doctors and pharmacists, let us first attempt to register a new nurse to our system to use for our user manual.
To begin this, select ‘Register’ at the bottom of the page.

Now, using the dropdown option menu, select Nurse as your role:



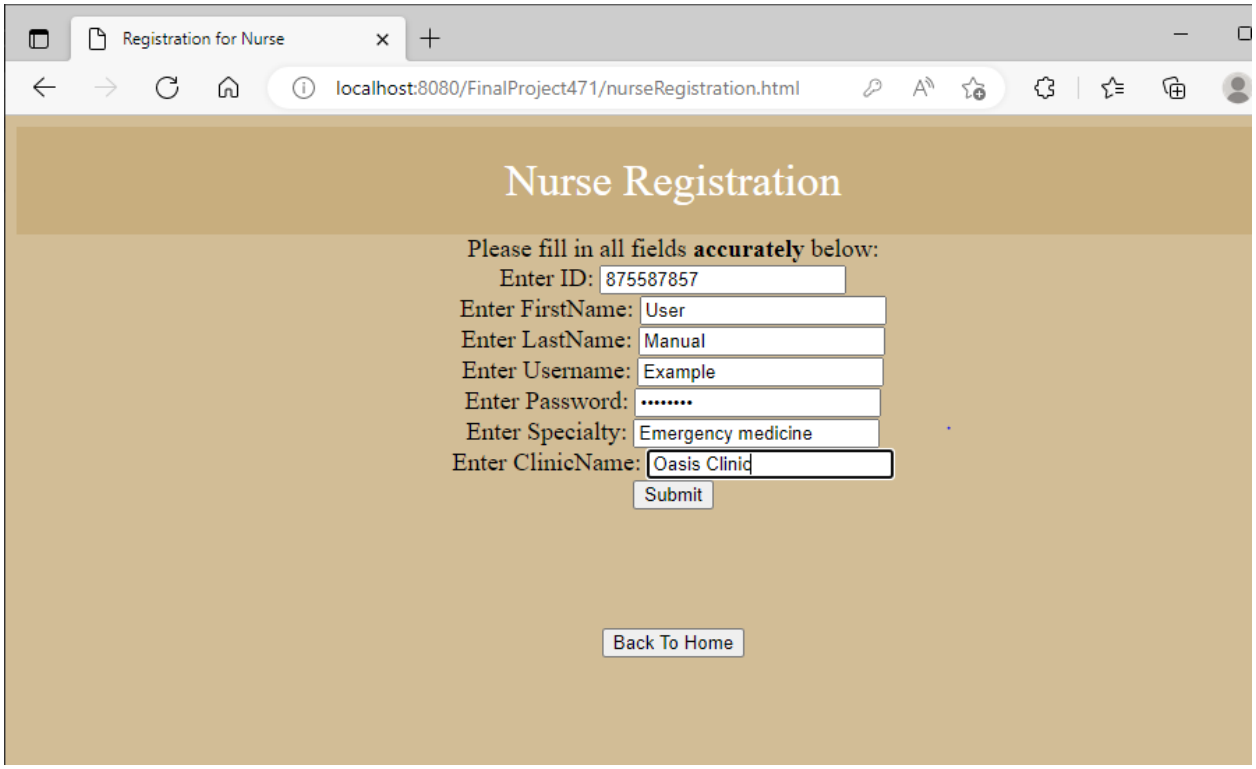
Role selection for registration. Here, we are choosing Nurse

Choose submit, and we now have this form presented to us:



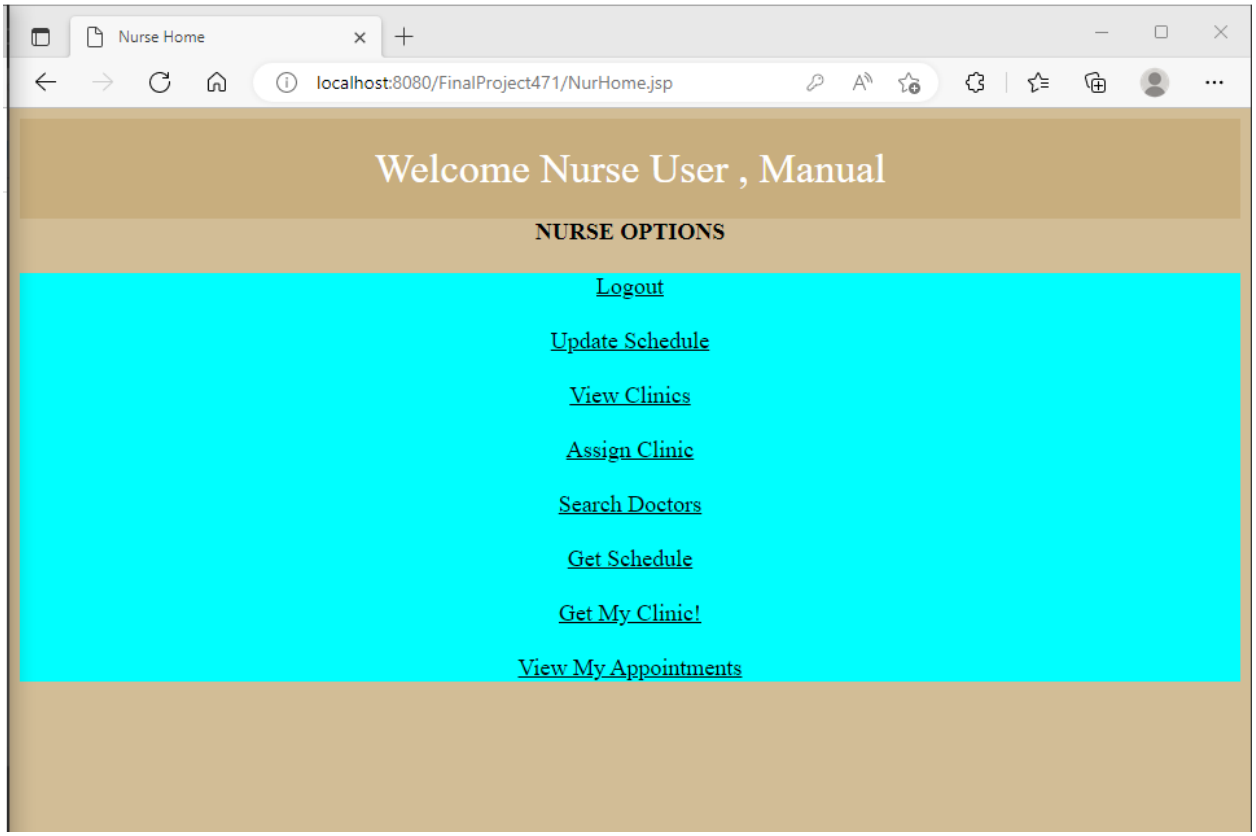
Nurse registration form

Now we can fill out this form to create our Nurse:



Filling in our sample Nurse registration form

Submitting this form yields the following:

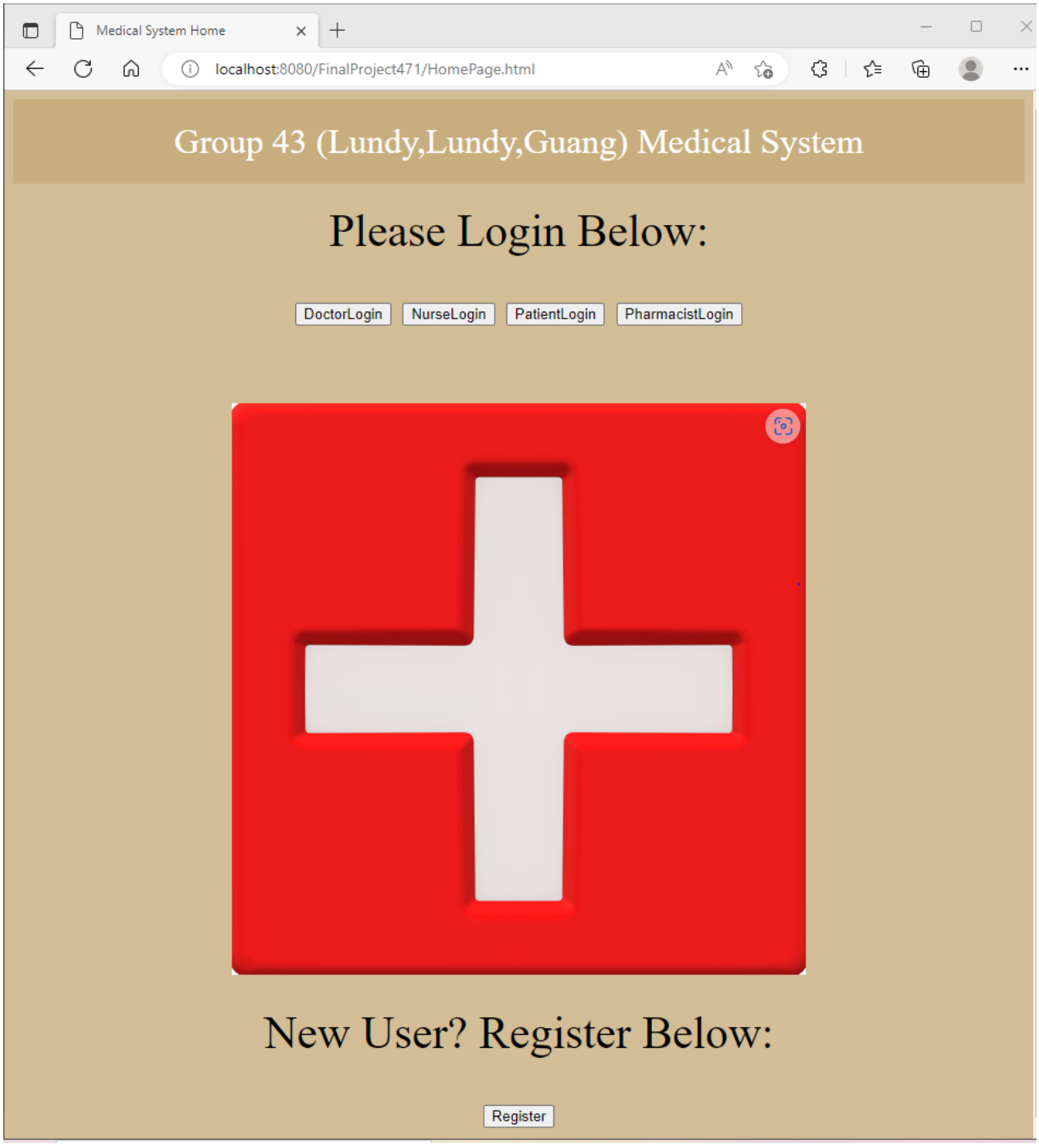


The Nurse Homepage for our user

This is the nurse homepage. Compare appendix u.1 for nurses with appendix u.16 to see the change in nurses in the database.

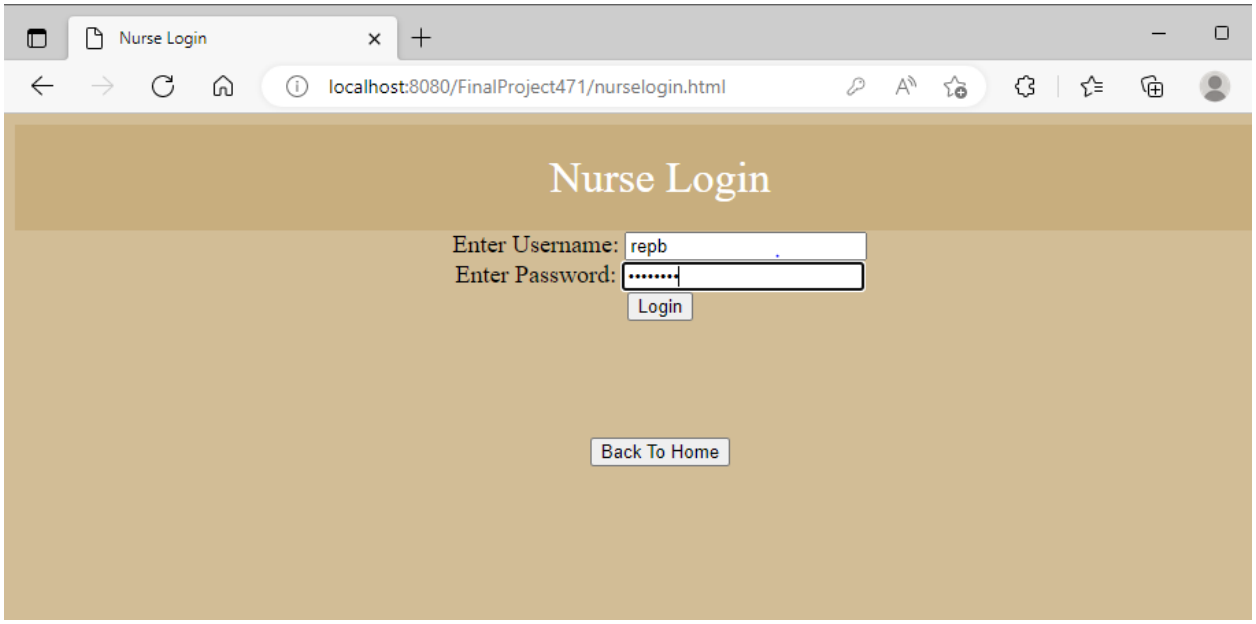
Now lets test the login/logout functionality of our system for nurses. We will show this process in the photos below. (1a, 1b) is a nonsense login (no nurse user match), (2a, 2b) is a wrong password login, (3a, 3b) is a successful login. (*a corresponds to input, b is the result*).

But first, we will press logout at the top, which brings us back to the nurse homepage. Then, we select ‘NurseLogin’ at the top of the webpage.

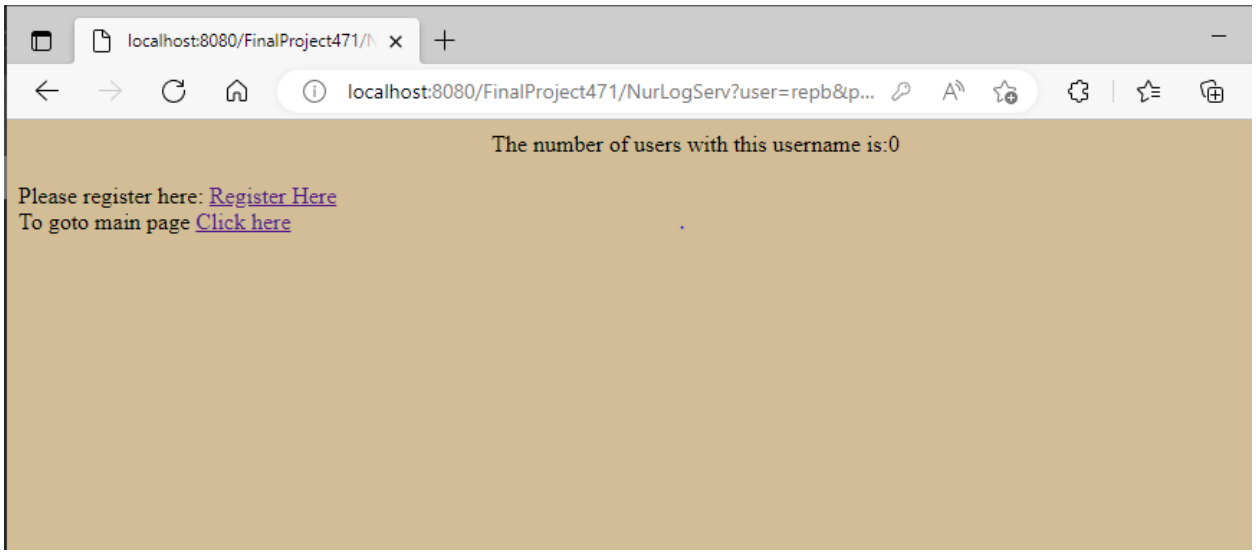


System homepage following nurse logout. Now we will attempt to login with 'NurseLogin'

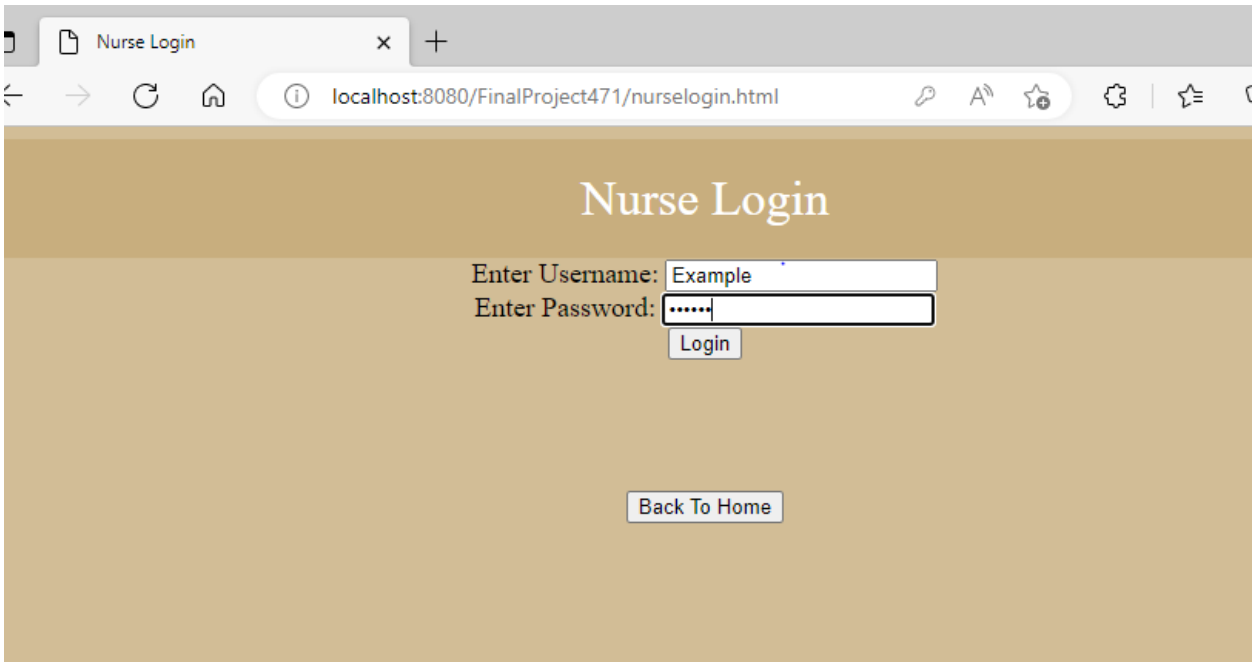
The following pictures depict the login situations outlined:



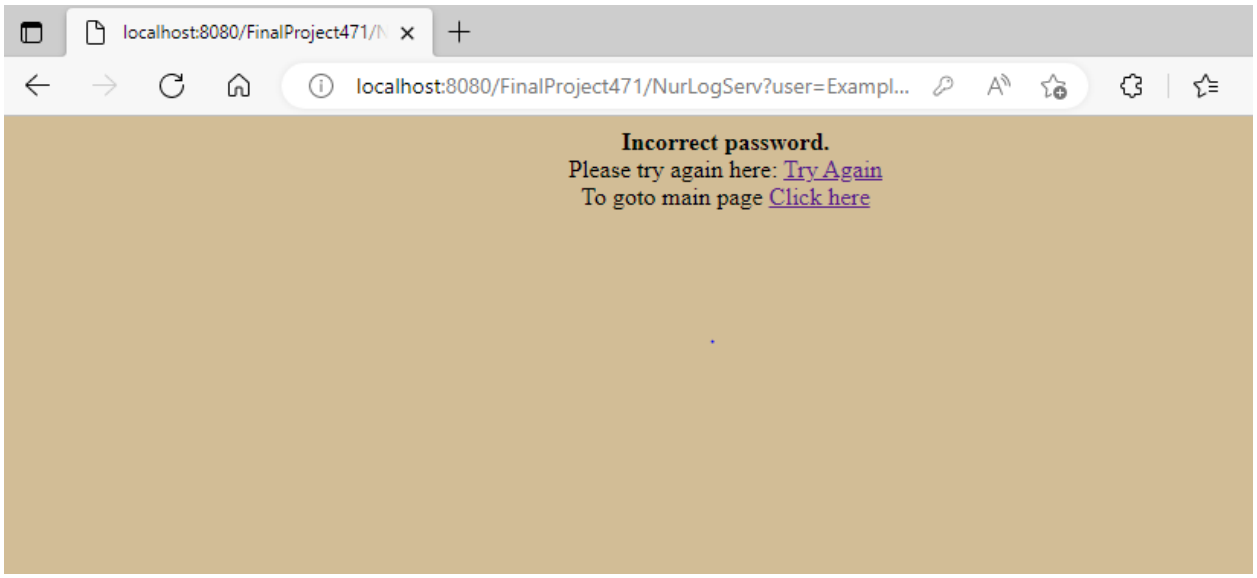
1A. Nonsensical login with nonexistent nurse



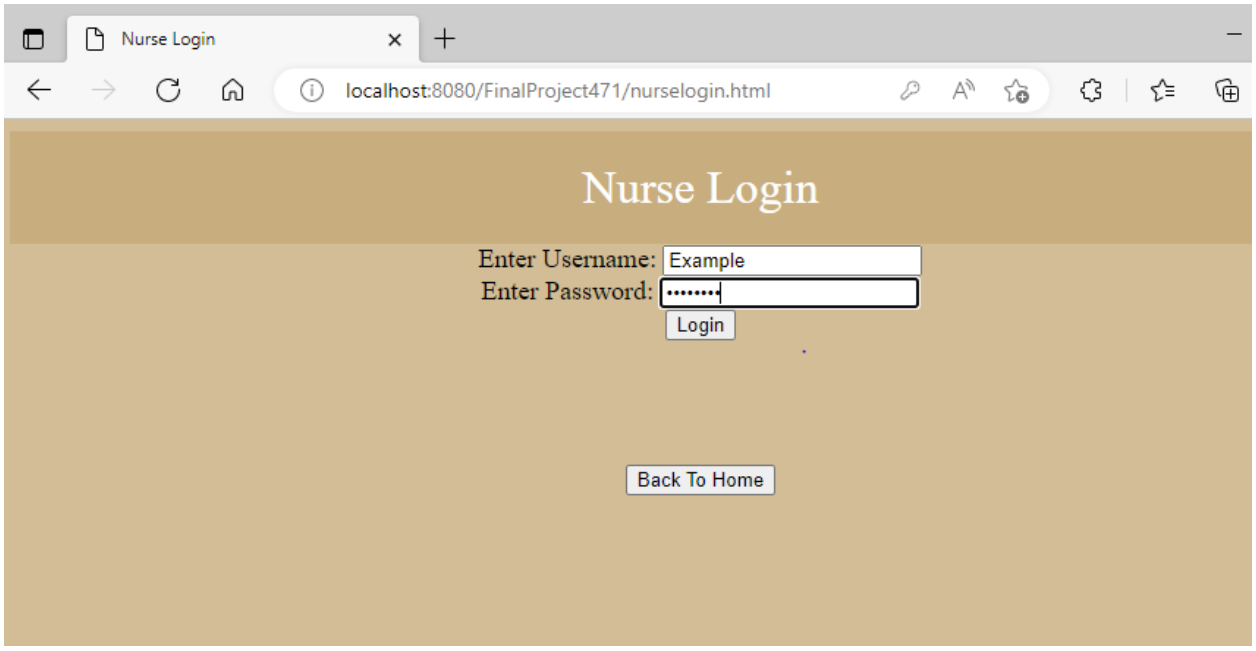
1B. The result (links to system homepage and to the registration page)



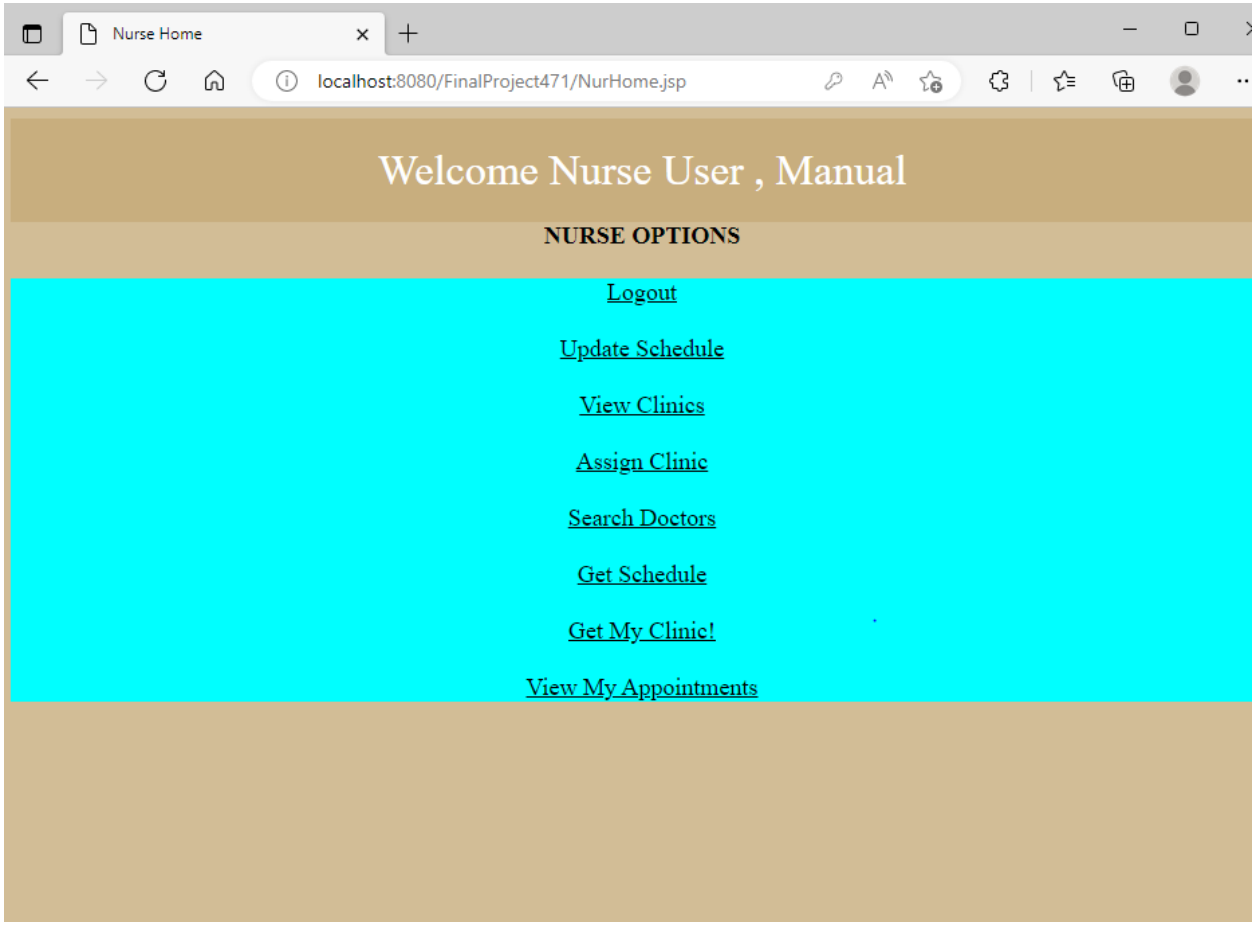
2a. Logging in with wrong password



2b. The result (link to retry login and link to system homepage)

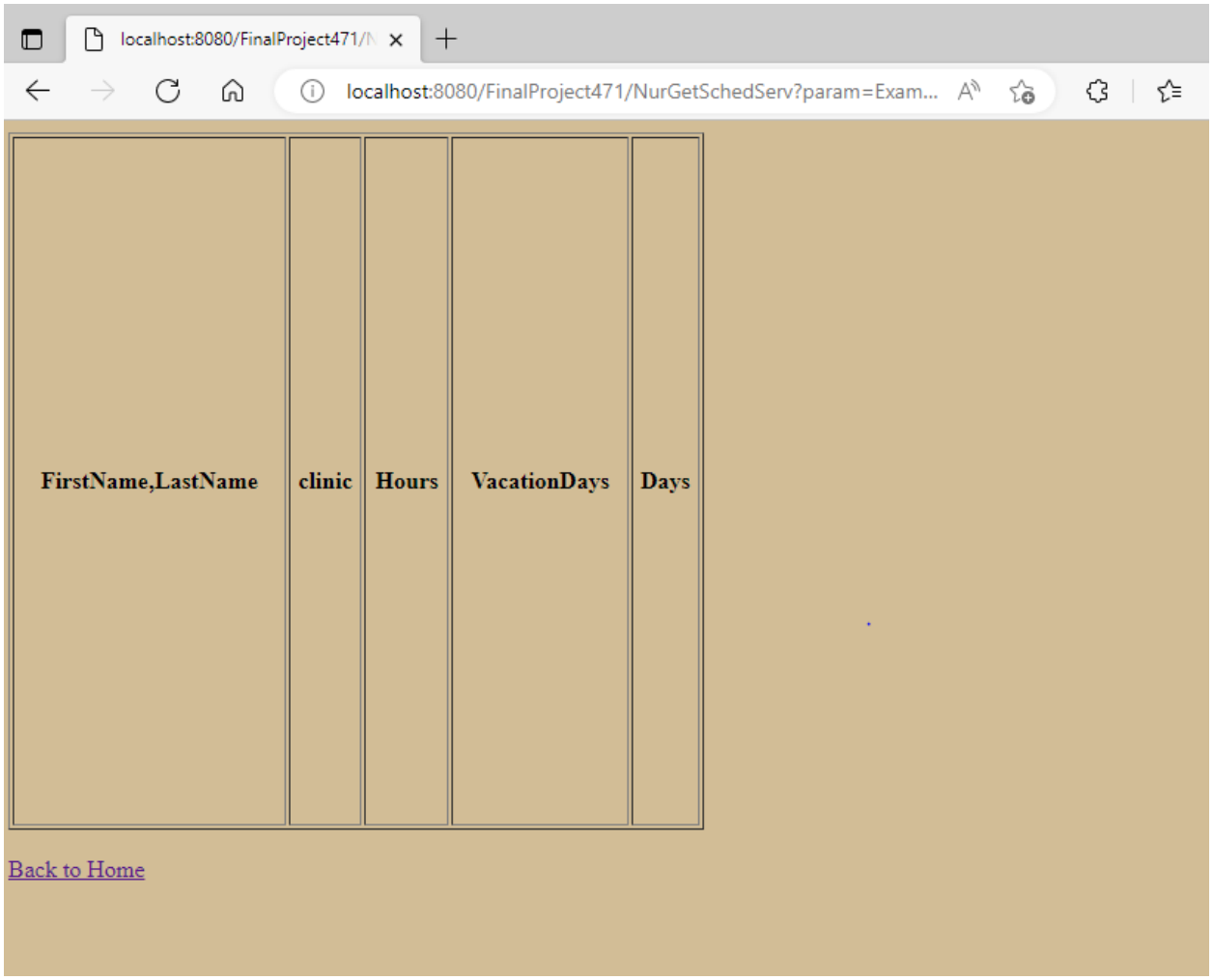


3a. Good login attempt



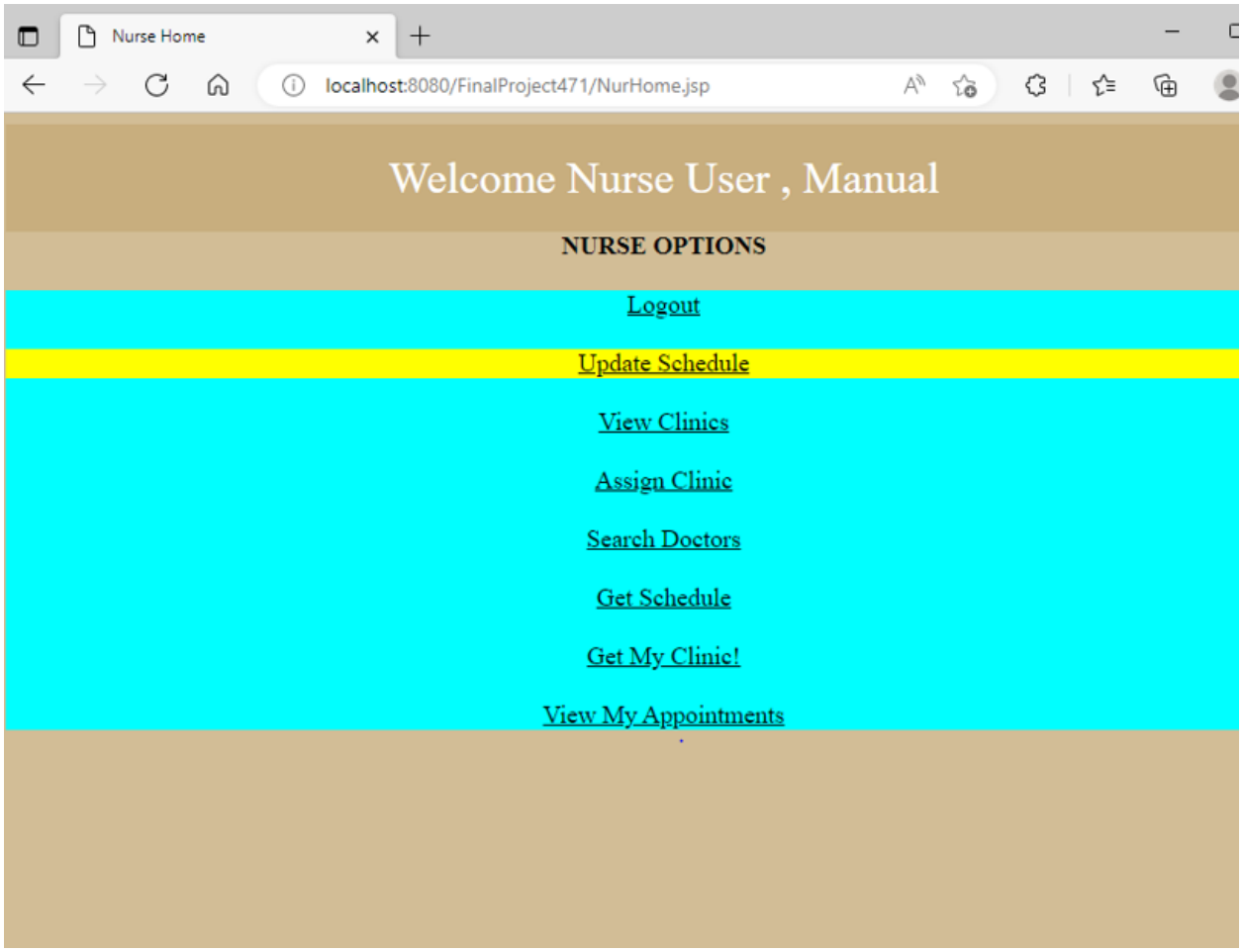
3b. The result - brought to Nurse's homepage

Now we have discussed the login functionality. We will continue by going through each nurse option in the order of the homepage, starting with update schedule. But first, it logically makes sense to show get schedule for a nurse who does not yet have a schedule.
Therefore, we will first select ‘Get Schedule’ (3rd from bottom).
This gives us this (expected empty since no schedule yet for our nurse example here):



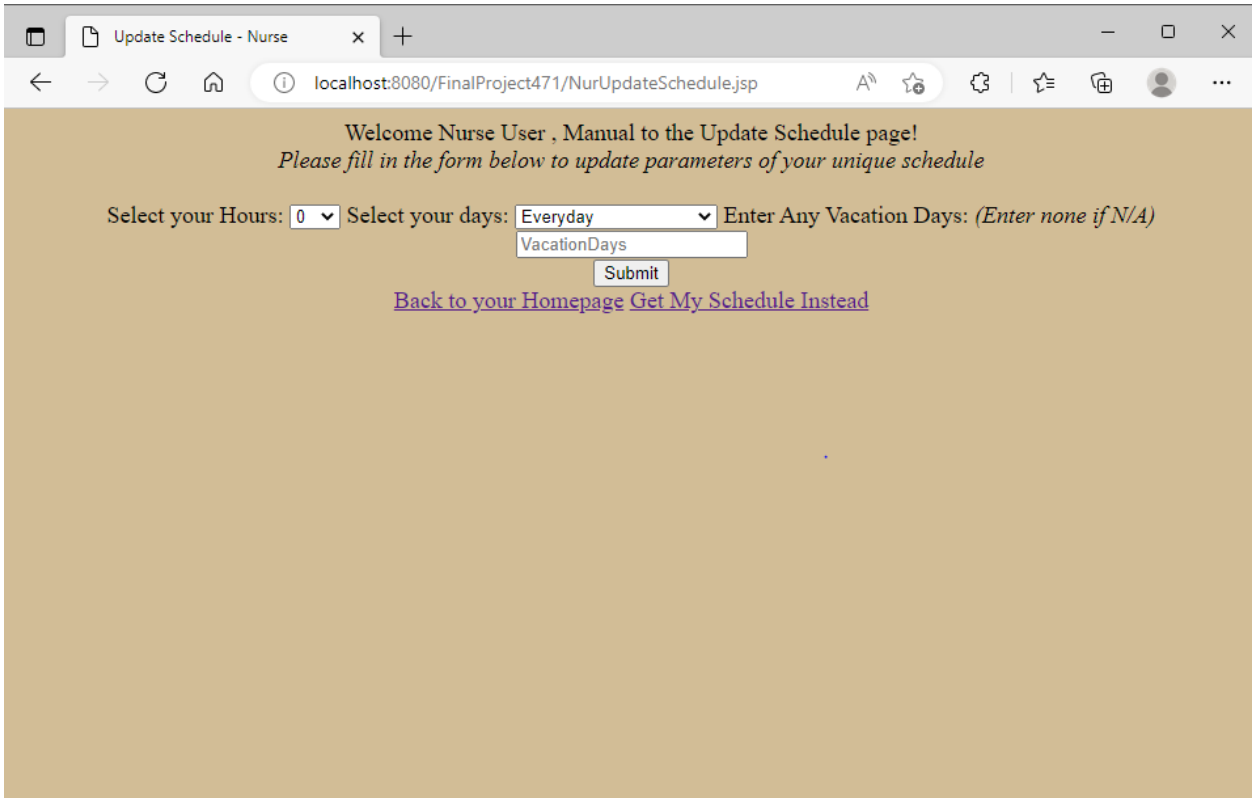
Empty schedule for nurse with no schedule yet (our example nurse)

Now, we can show update schedule and then view the schedule again.
So lets press Back to Home to get back to the Nurse homepage and then select the second option from the top: Update Schedule.



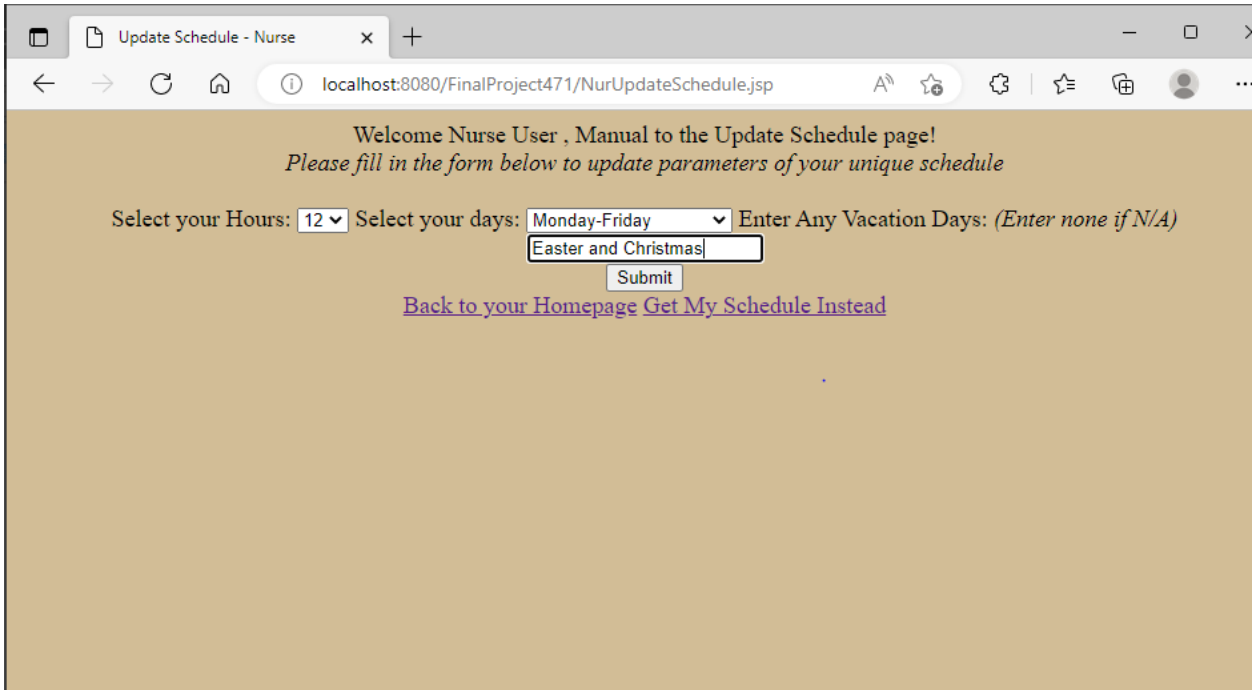
Nurse homepage. Highlighted: Update schedule option

Once this option is selected, we have this page:



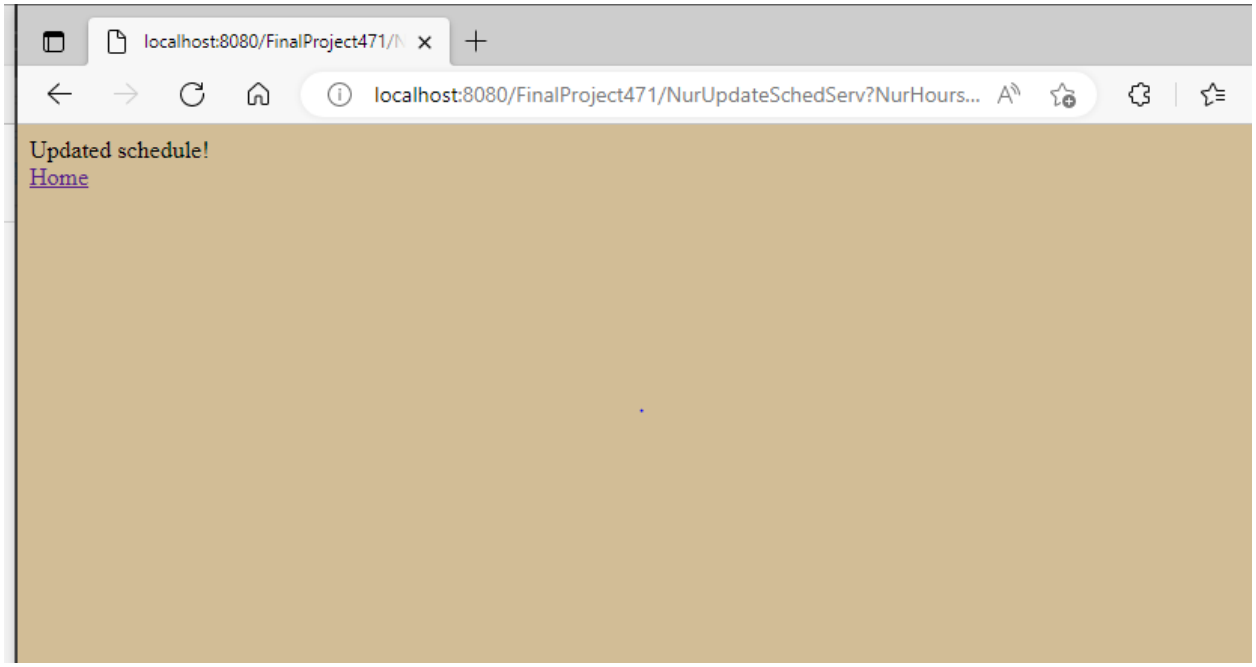
Update schedule page

Now we see a dropdown menu for selecting hours, days and vacation days. Much like in the doctor update schedule seen earlier in this manual, hours contains options 0-24, and days ranges from everyday to Monday to Sunday with every combination in between. Vacation Days, of course, is a required/empty protected text field.
Also note the two links underneath: Back to Homepage, which brings the user back to their nurse homepage, as well as Get my Schedule instead, which will go to the get my schedule we saw before for that user.
Lets fill in our example with sample values as such:



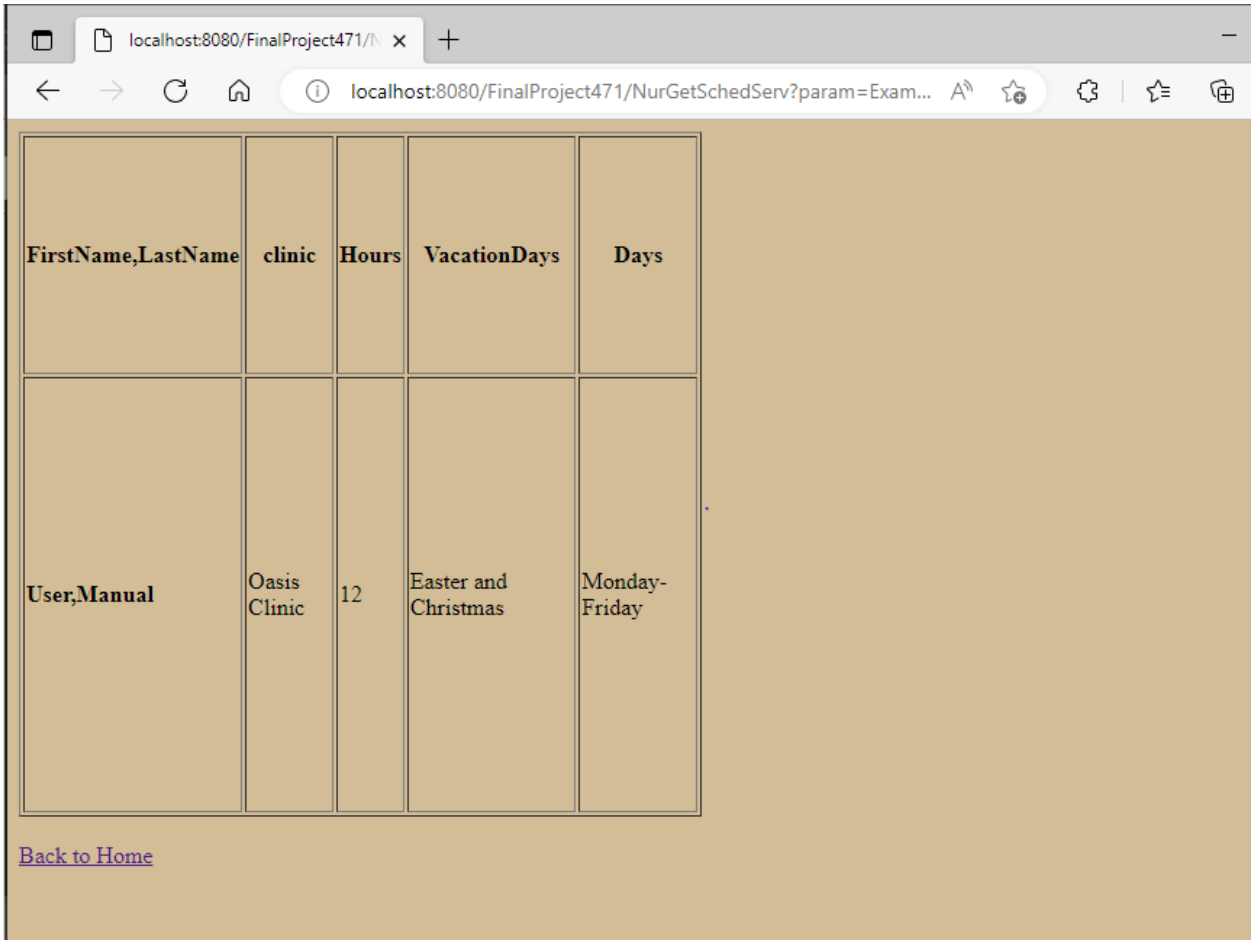
Sample values for update schedule for Nurse

Submitting this form gives us this:



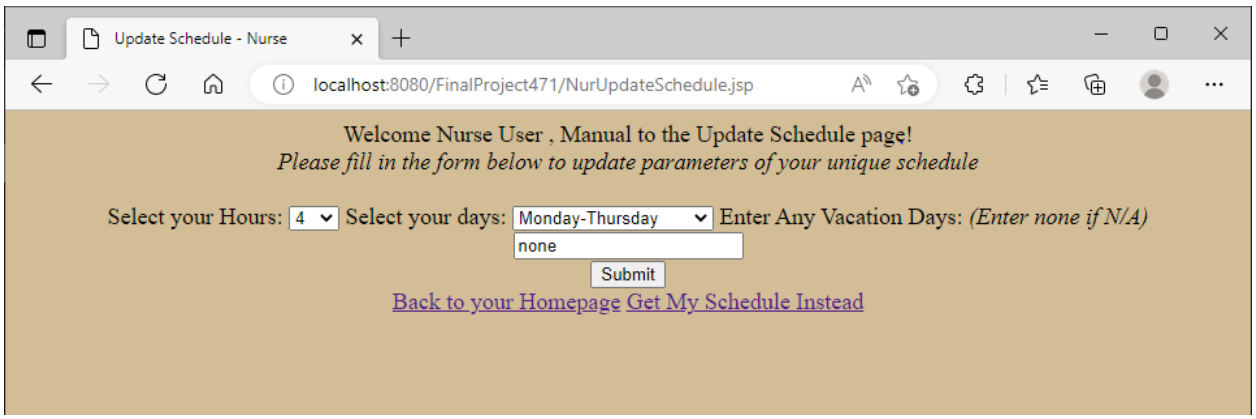
Updated schedule message

Now we can refer to appendix u.17 to see the change to nurseschedule for that user.
Now that we have a schedule for this user lets select Home, then select get schedule again to see our newly created schedule.

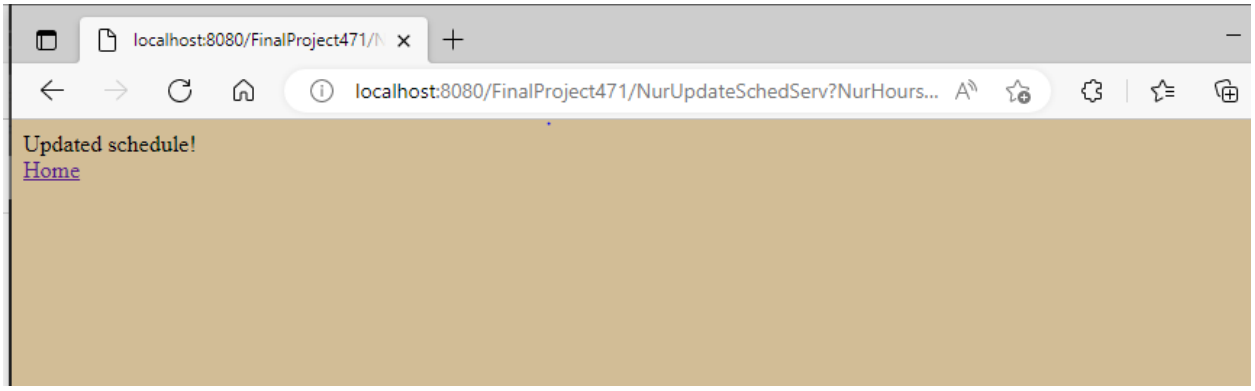


After going back to the homepage and then pressing get schedule, we see our new schedule

Now lets quickly overwrite this schedule to show how our system updates an existing schedule. To do this, go back home and select update schedule. Then we will fill it in with this example:

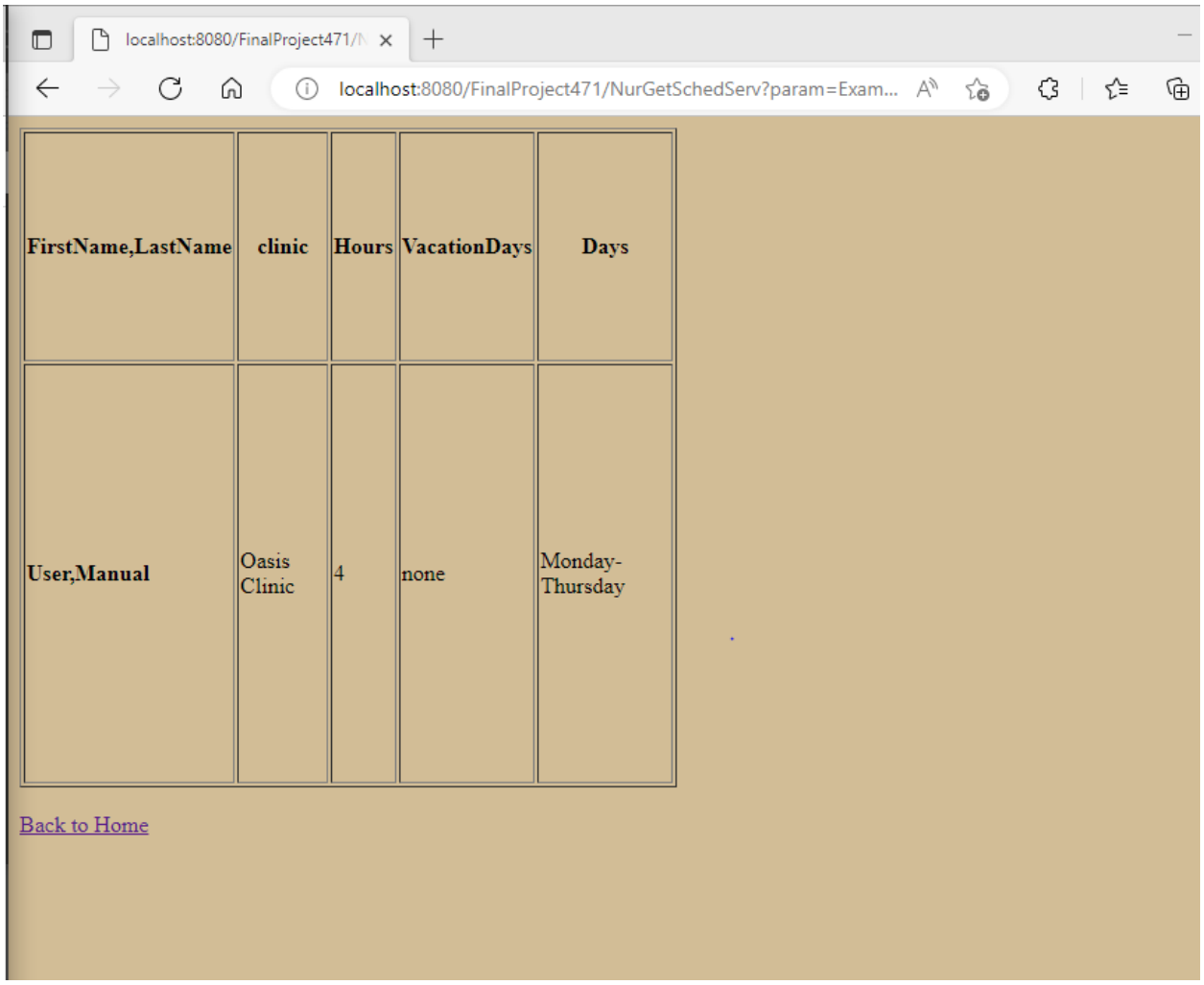


New form for the same Nurse user to overwrite their schedule



Submitting that example form above result

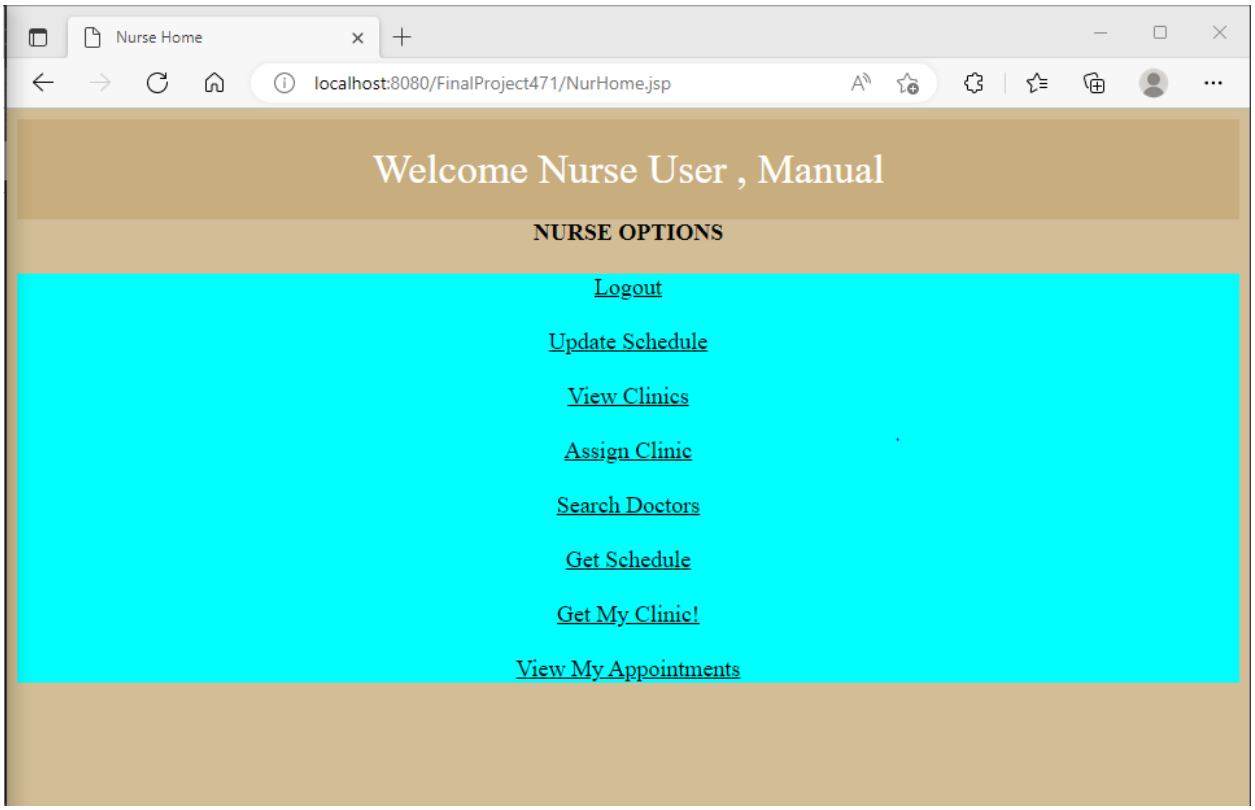
Now, we can go back and press get schedule again to see the changes (which overwrites the existing schedule).



Get Schedule after update

See appendix u18 to view database change here.

Now we can go back to the Nurse homepage to look at the other Nurse functionality.

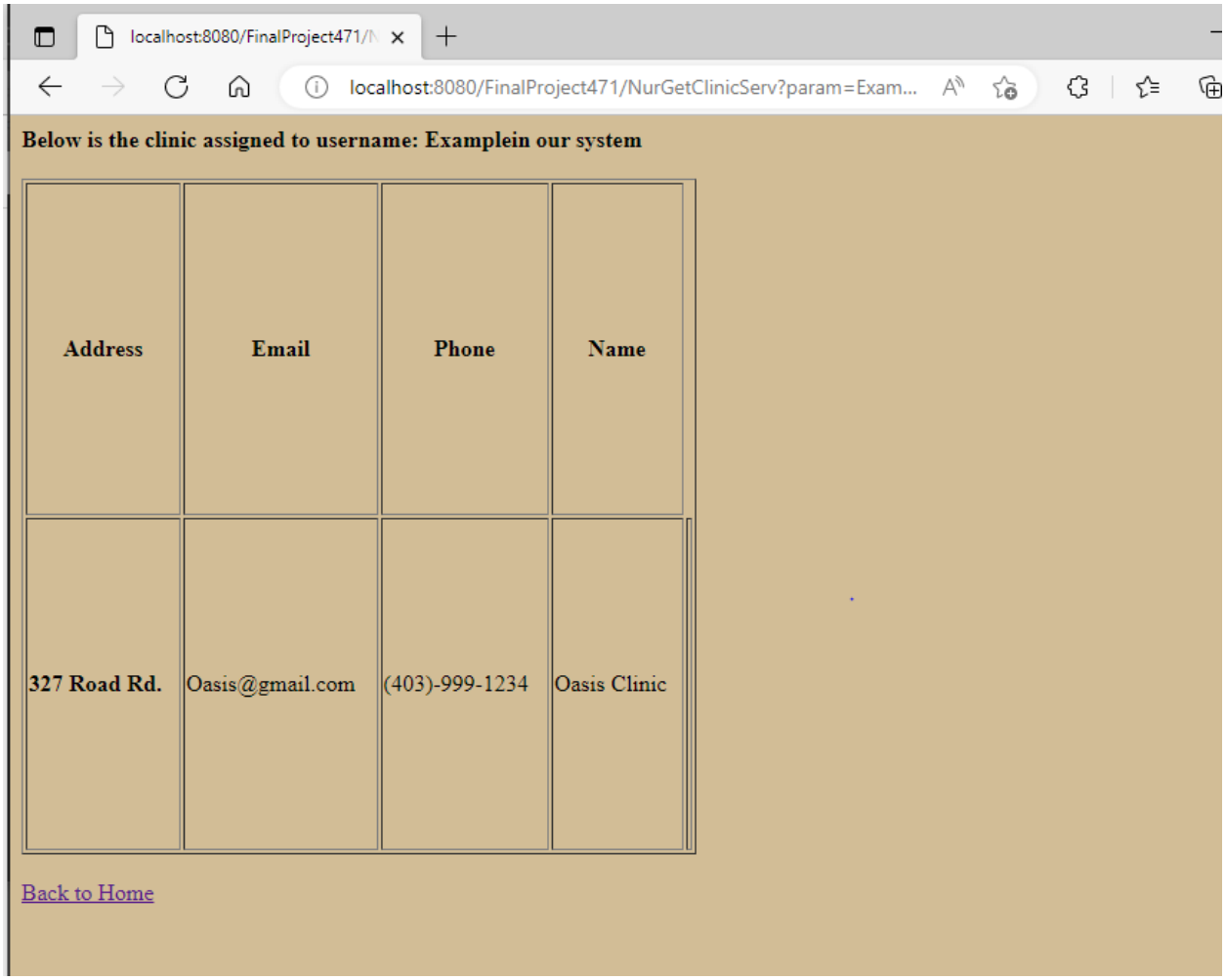


Nurse Homepage

Now we can discuss the third option: View Clinics. Logically, in order to begin this discussion we must first discuss the option: Get My Clinic (second from bottom) and partially discuss Assign clinic (we will discuss fully soon).

First, we will show the get my clinic option, which will retrieve the clinic that this nurse works in.

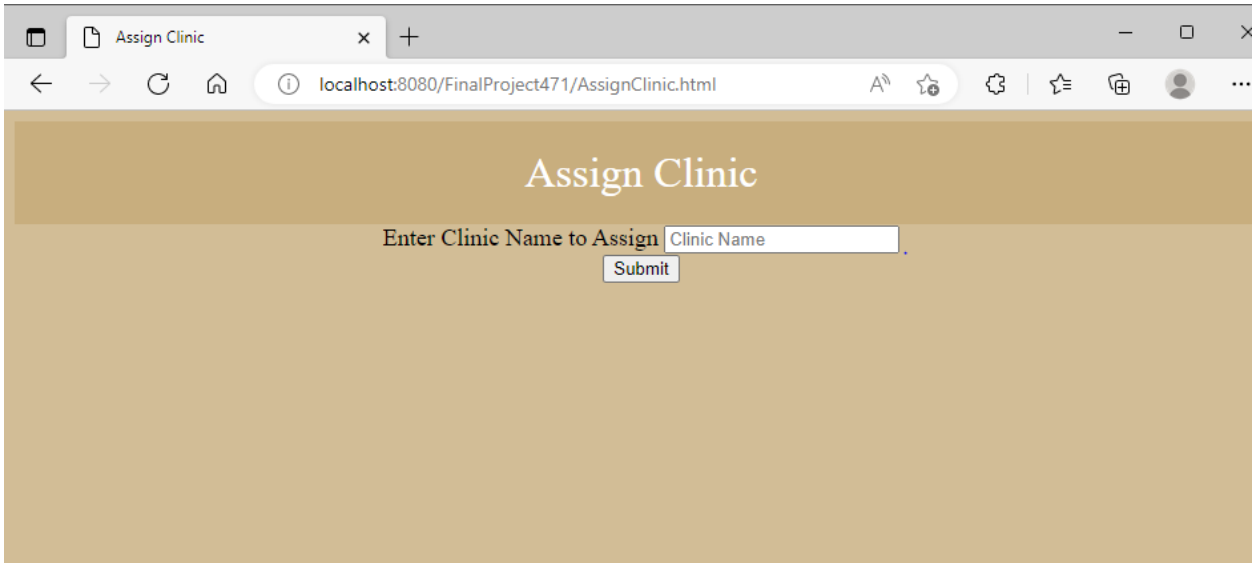
Selecting this option (second last option) gives us:



Get My Clinic for our example nurse

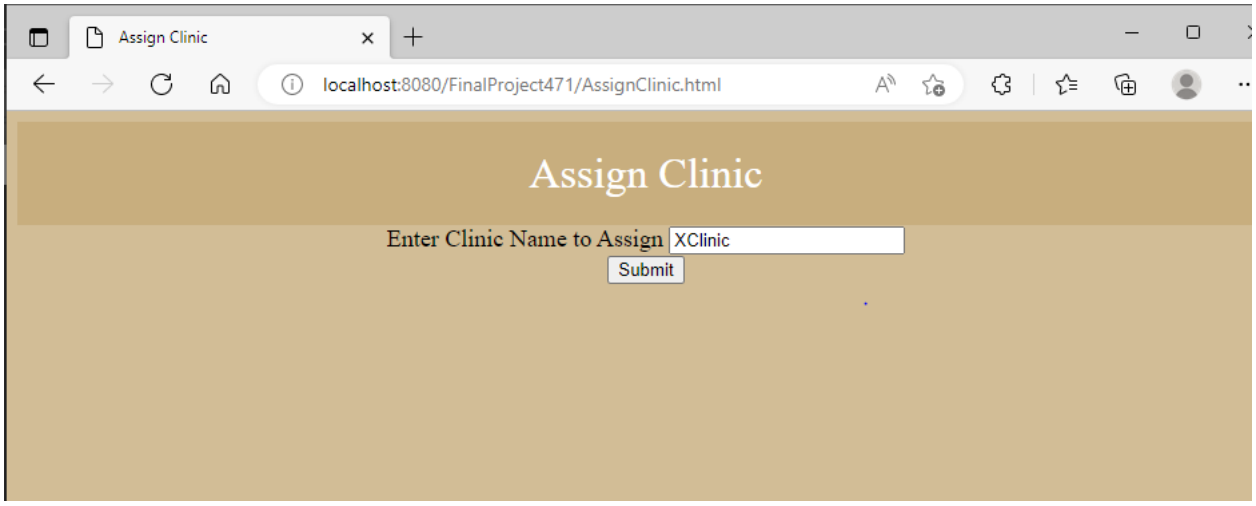
As we can see, this option retrieves the clinic information of the clinic that this nurse works in. Now we can go back to the Nurse homepage and discuss Assign Clinic and view clinics. First, select ‘Assign Clinic’ on the homepage.

Once selected, we see this:

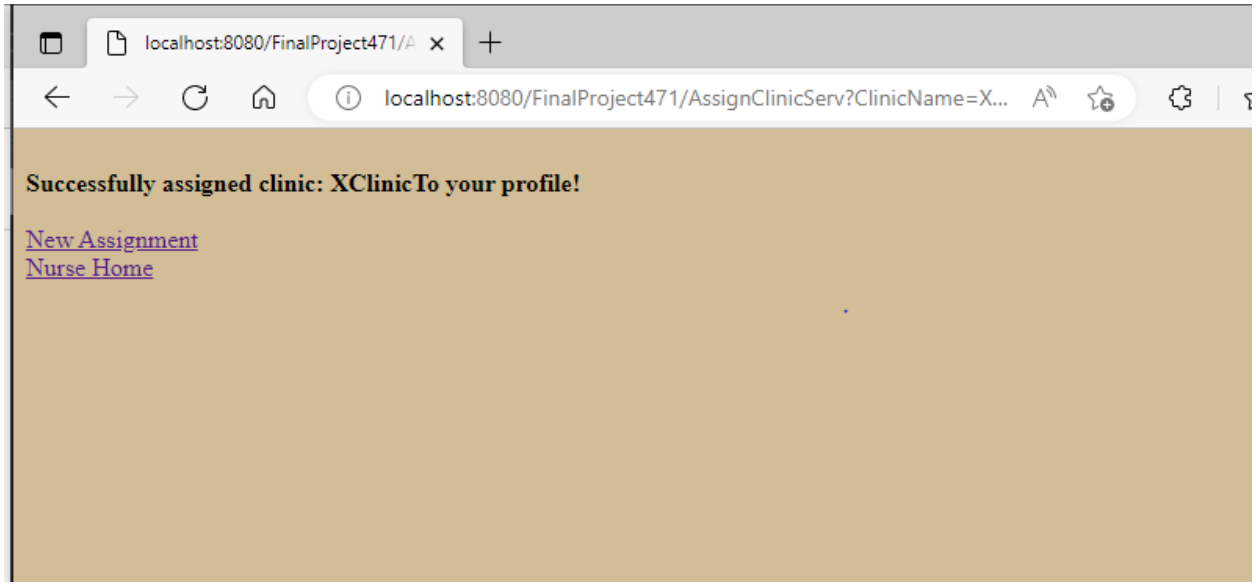


Assign Clinic Page

Now we simply enter the clinic name to assign to our nurse, and the change will be reflected (appendix u19).

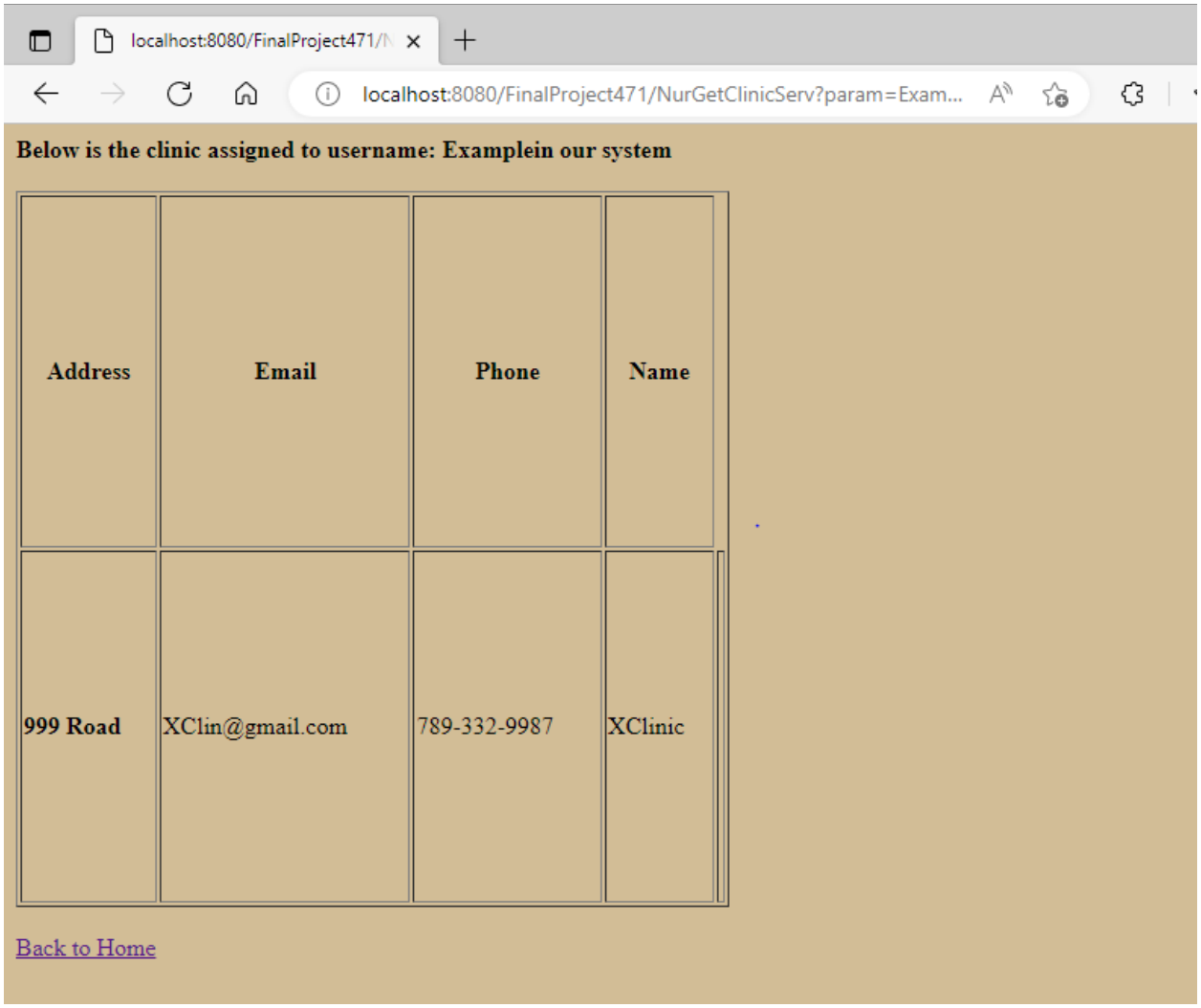


Assigning our nurse to XClinic



The result. Now, we can go back home or we can go back to the assignment page to assign our nurse to a different clinic again.

Now when we go back to the homepage and select get my clinic we see the change reflected:

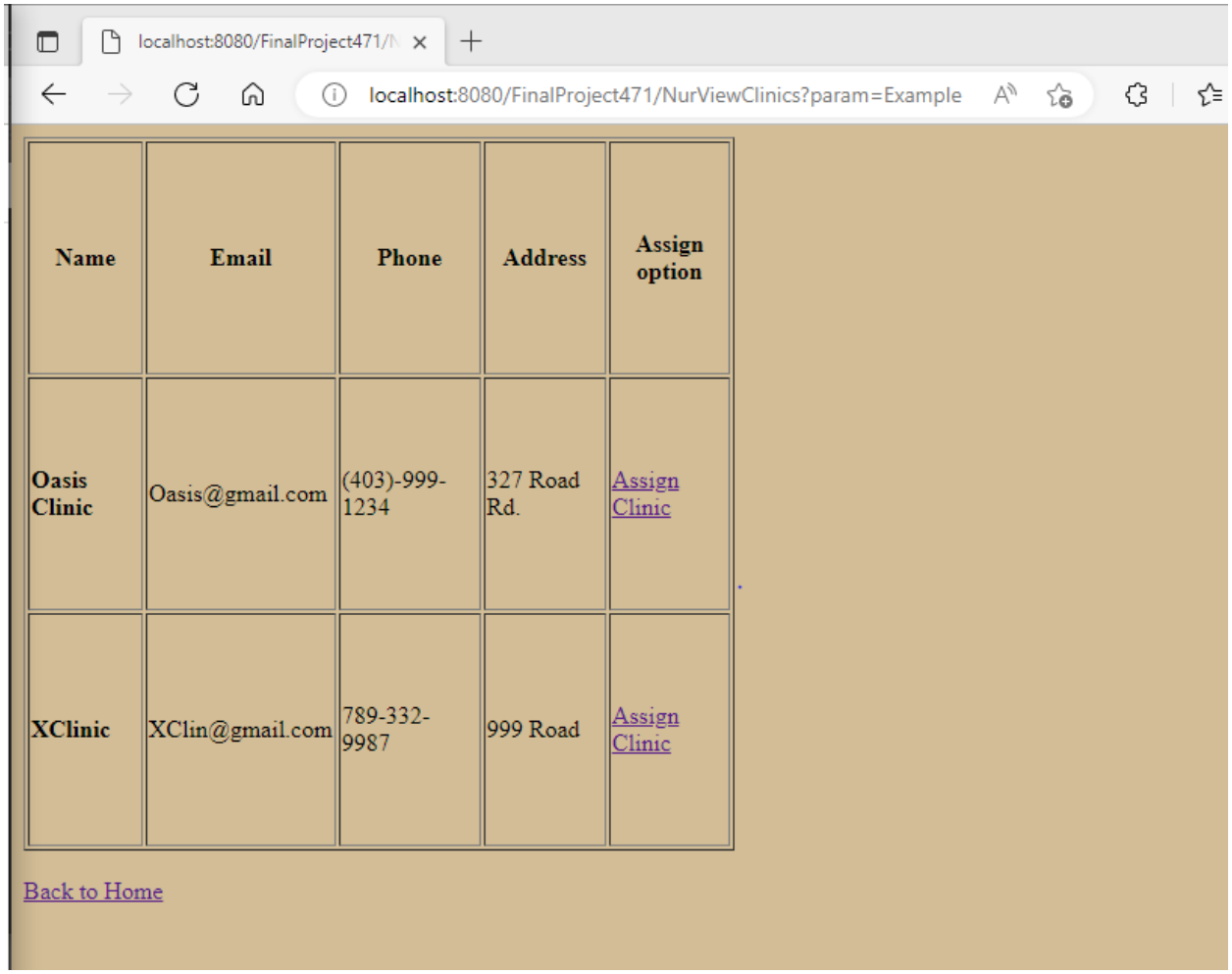


New Get My Clinic with assigned clinic shown

Assigning the nurse to the same clinic does nothing special - ie. it updates from XClinic to XClinic (for example), and thus no real change is made. Since it is an update, this is fine and suitable for our system, since there won't be numerous records rather just one for that nurse.

Now that we have seen get my clinic as well as assign clinic, we can finally discuss the third option on the nurse homepage: view clinics.
To start, we'll press View Clinics from the Nurse Homepage.

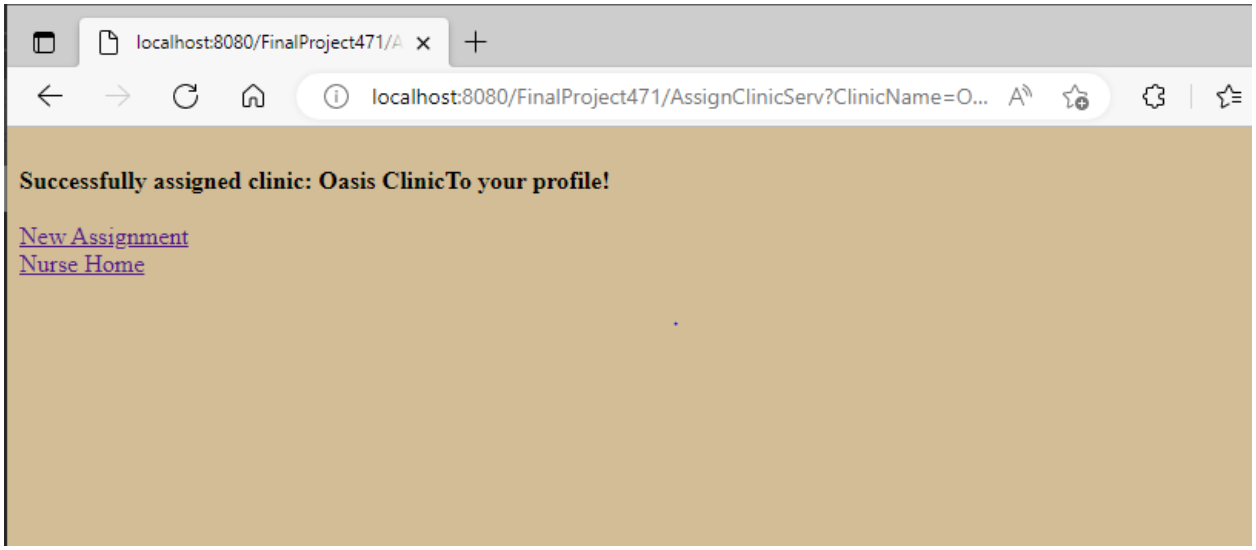
Doing so results in this:



Result of pressing View Clinics from Nurse Homepage

As you can see, this simply retrieves every clinic registered in our system with the information for each clinic. Note that the final cell in the table includes a link to Assign clinic. Pressing this link simply does the same functionality as the Assign Clinic option discussed before. However, this link takes out the need to have a text form, instead assigns that row's clinic immediately to the nurse.

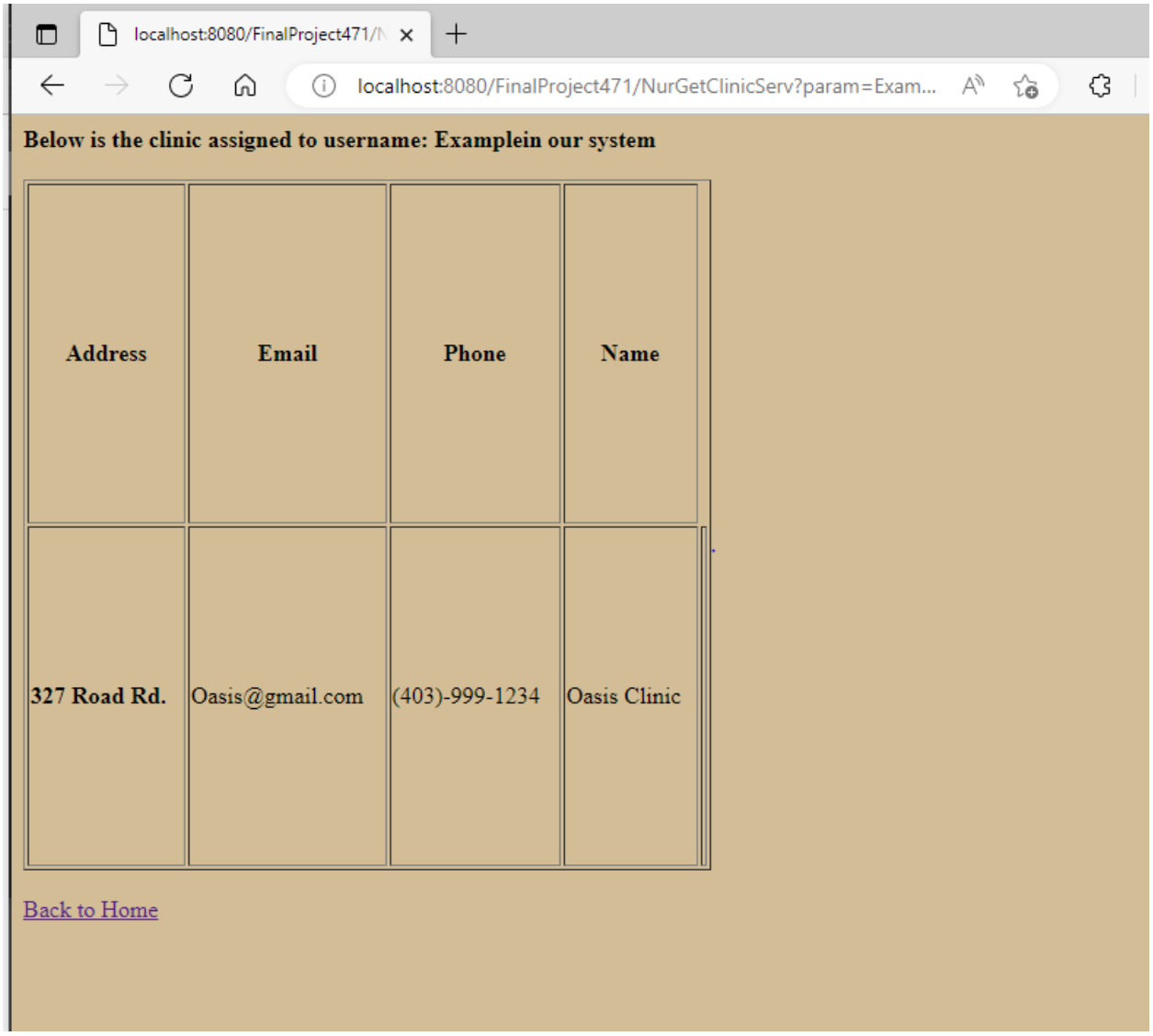
For example, lets press assign clinic for the first row to assign Oasis Clinic to this nurse who is currently at XClinc.



Result of pressing Assign Clinic link for Oasis Clinic row

Seek appendix u20 to see the database change.

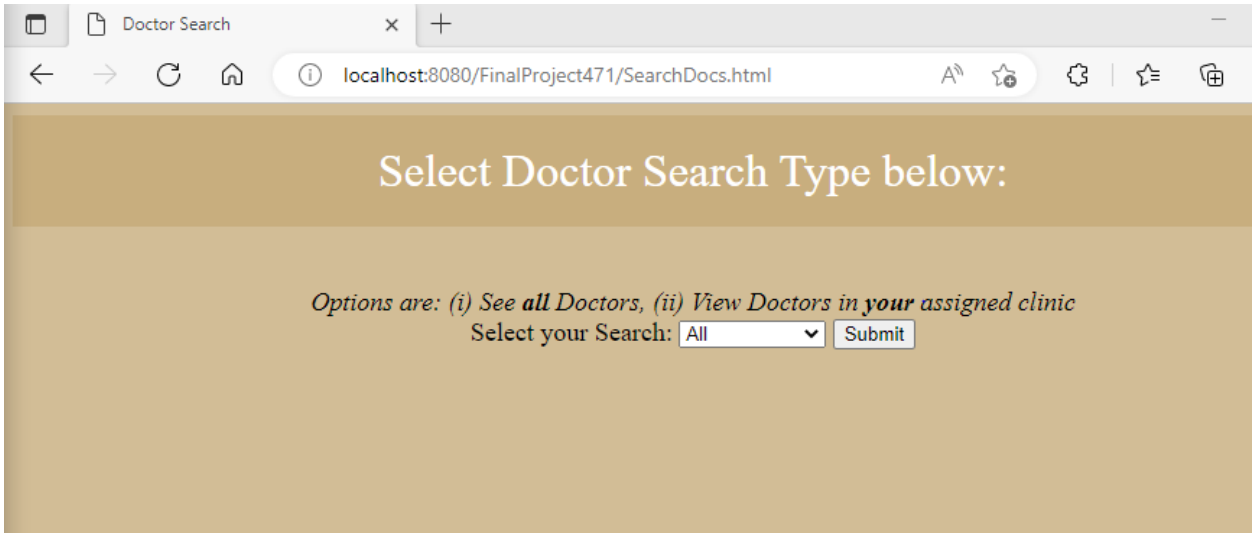
Also, if we go back and press get my clinic we observe:



Get My Clinic result after assigning Oasis Clinic

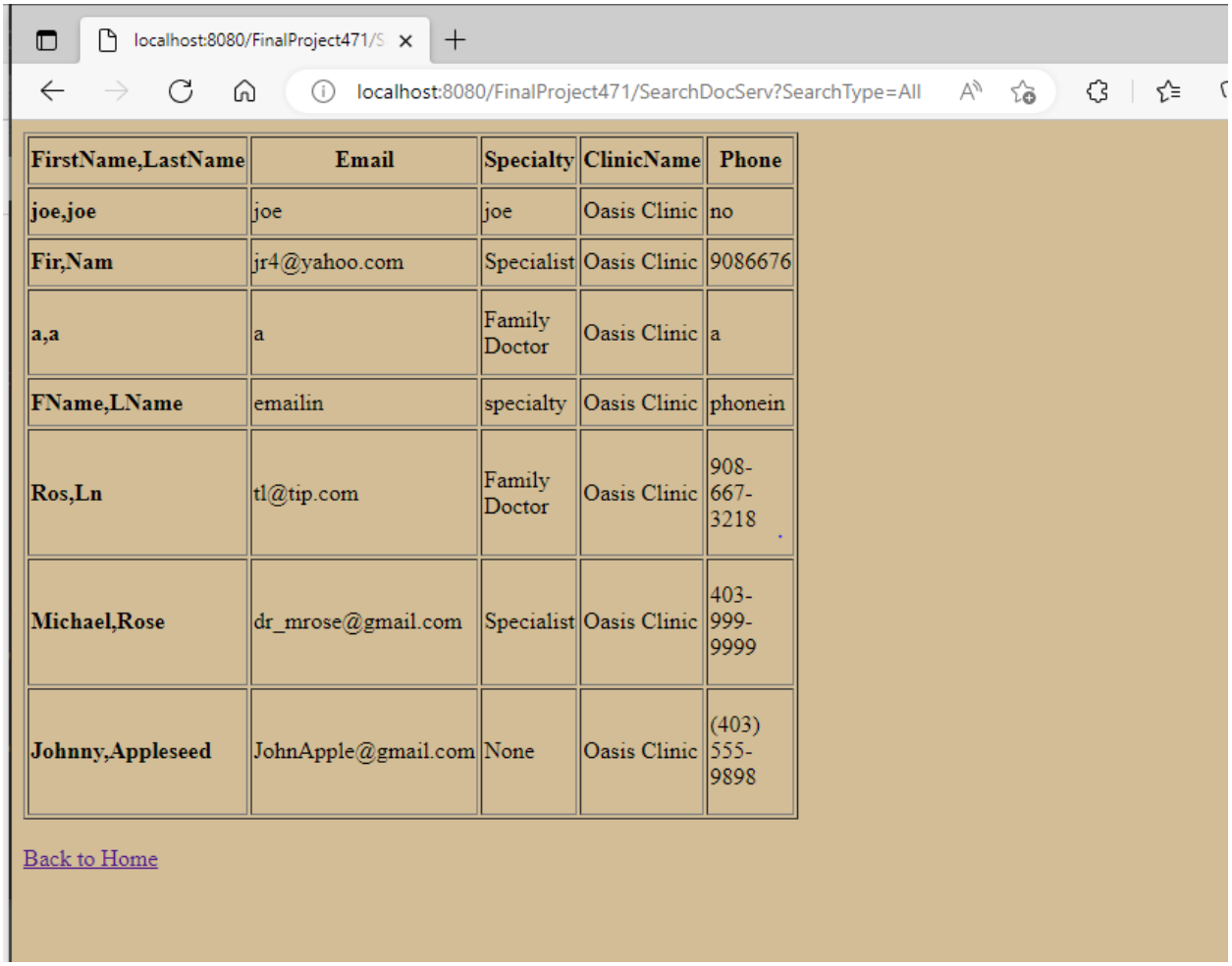
The final two options left for Nurse are: Search Doctors and View My Appointments from the Nurse Homepage.
First, lets go through the Search Doctor option from the homepage.
To do this, go back to the nurse homepage then select Search Doctors.

Now we will see this:

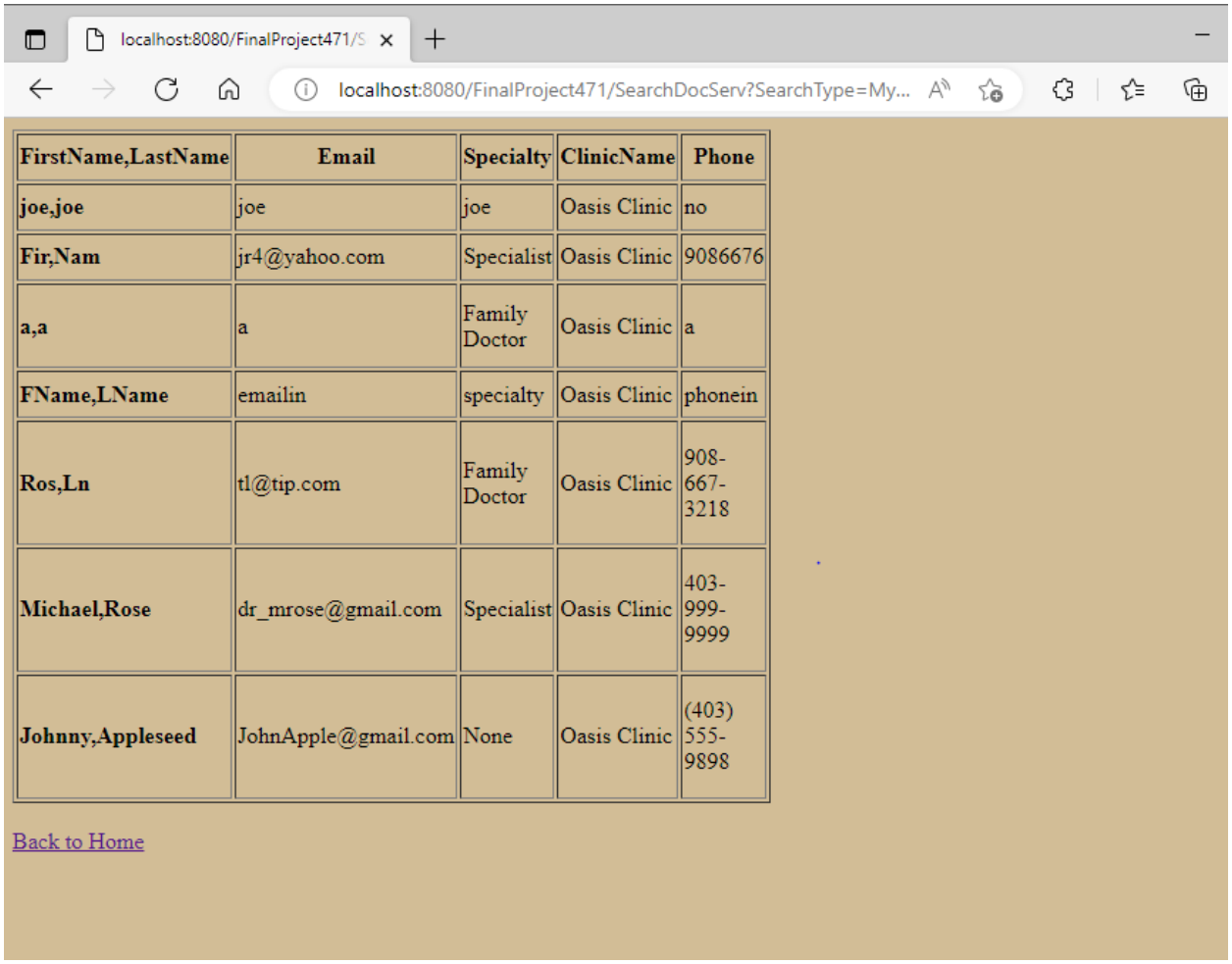


The Doctor Search form

Here, we have a dropdown menu which has 2 options: (1) all doctors or (2) MyClinicOnly.
Option (1) allows you to see all doctors and their registered information.
Option (2) allows you to see only the doctors that work in the same clinic as the logged in nurse.
Below is the option(1) and option(2) in pictures:
Note that at this point all doctors are in Oasis Clinic and none are in XClinic which is reflected in the photos below. Check the appendix to verify this fact. (appendix u1 then look for clinics table).



Option (1): Choosing 'all' for doctor search.

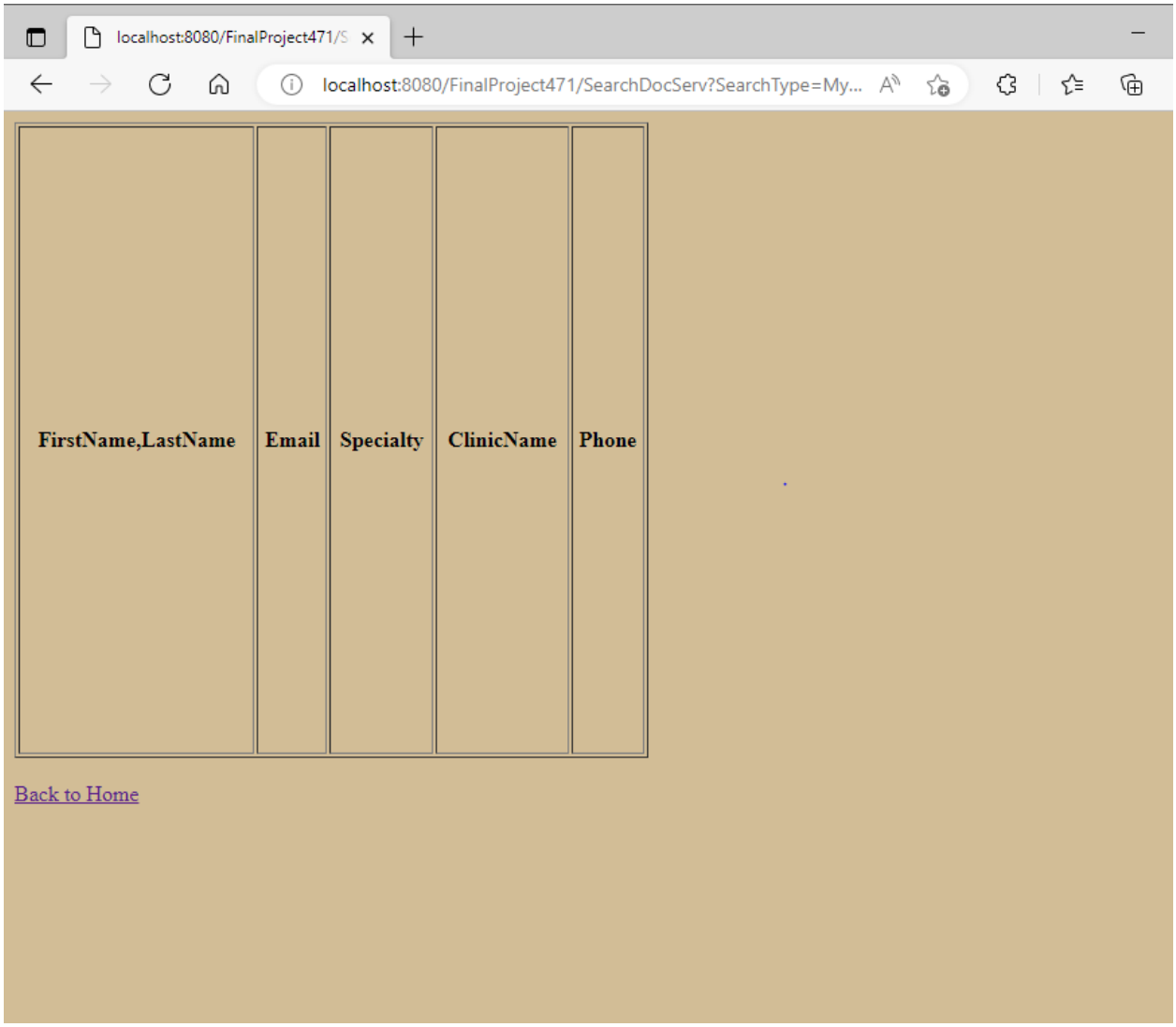


FirstName,LastName	Email	Specialty	ClinicName	Phone
joe,joe	joe	joe	Oasis Clinic	no
Fir,Nam	jr4@yahoo.com	Specialist	Oasis Clinic	9086676
a,a	a	Family Doctor	Oasis Clinic	a
FName,LName	emailin	specialty	Oasis Clinic	phonein
Ros,Ln	tl@tip.com	Family Doctor	Oasis Clinic	908-667-3218
Michael,Rose	dr_mrose@gmail.com	Specialist	Oasis Clinic	403-999-9999
Johnny,Appleseed	JohnApple@gmail.com	None	Oasis Clinic	(403) 555-9898

Back to Home

Option (2): Doctors search with 'MyClinicOnly' selected instead.
Note that at this point in our sample database every doctor is in Oasis Clinic (which our nurse works in), and thus these two pictures are equal.

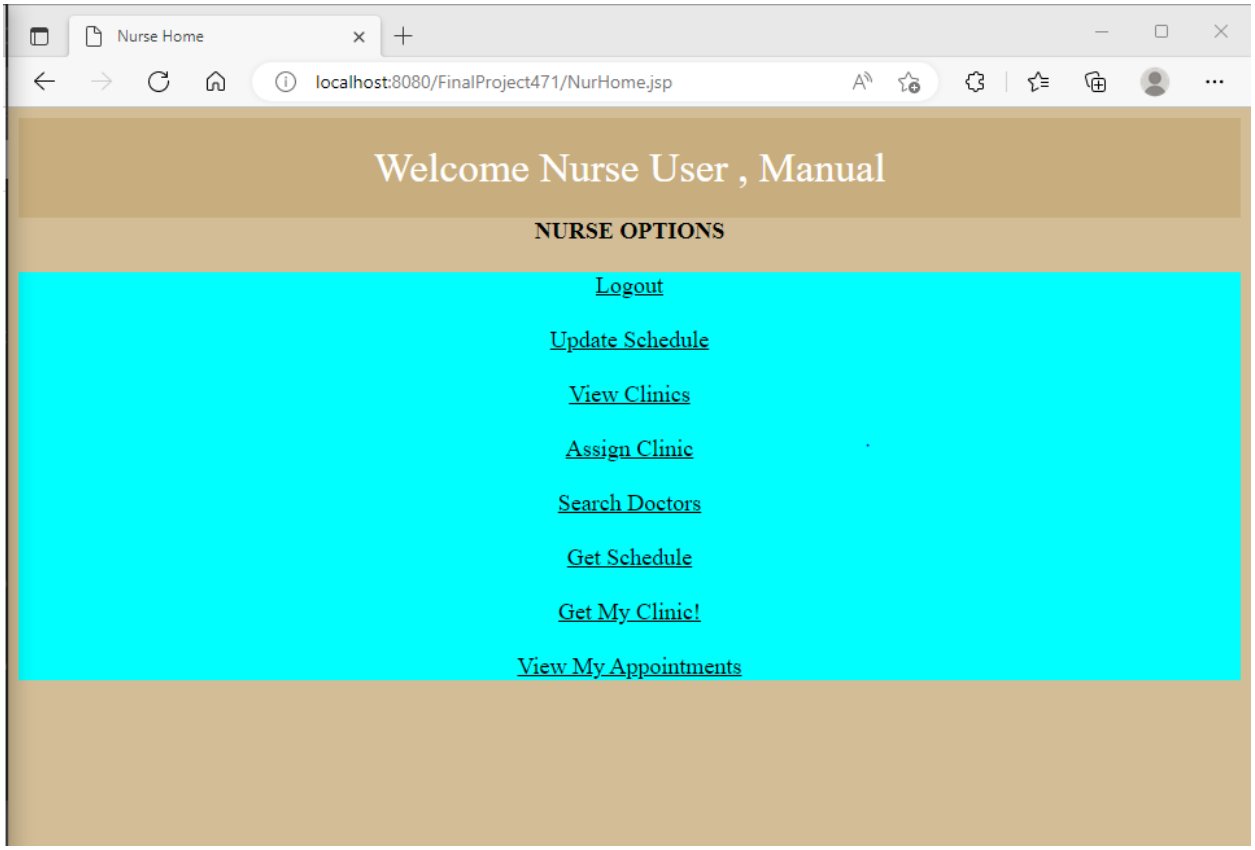
Now, if we go home and re-assign this nurse to XClinic and then choose Doctor Search with MyClinicOnly, we will see:



Search Doctors in MyClinicOnly after re-assigning to XClinic, which currently has no doctors

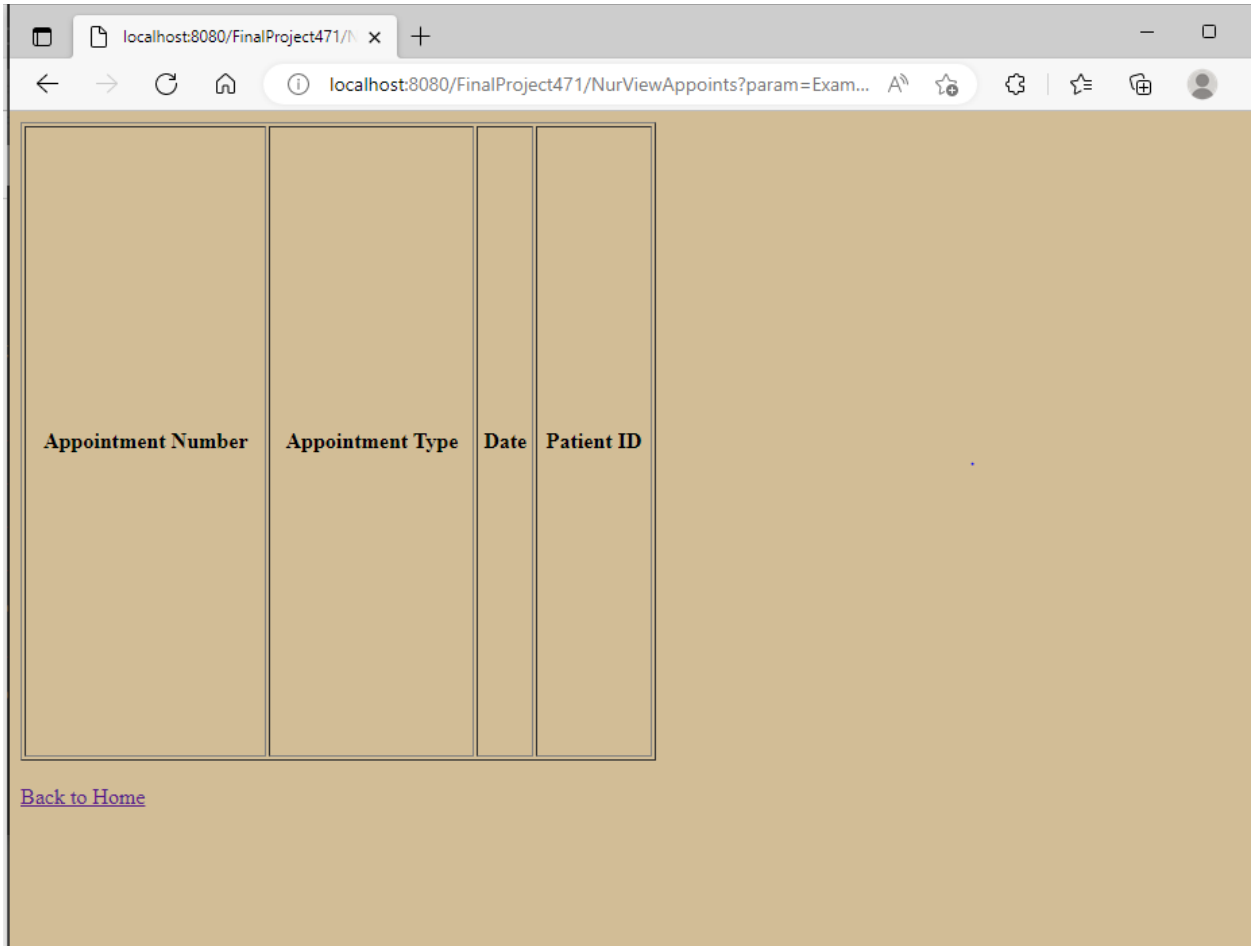
And, as expected, this is empty since XClinic has no doctors in our sample database (view appendix u1 for doctor table to verify this fact).

Now we can finally explore the final option for Nurse: View My Appointments.
To do this, let us go back to the Nurse Homepage and select the final link option:



Nurse Home - we will select the last option (view my appointments) now

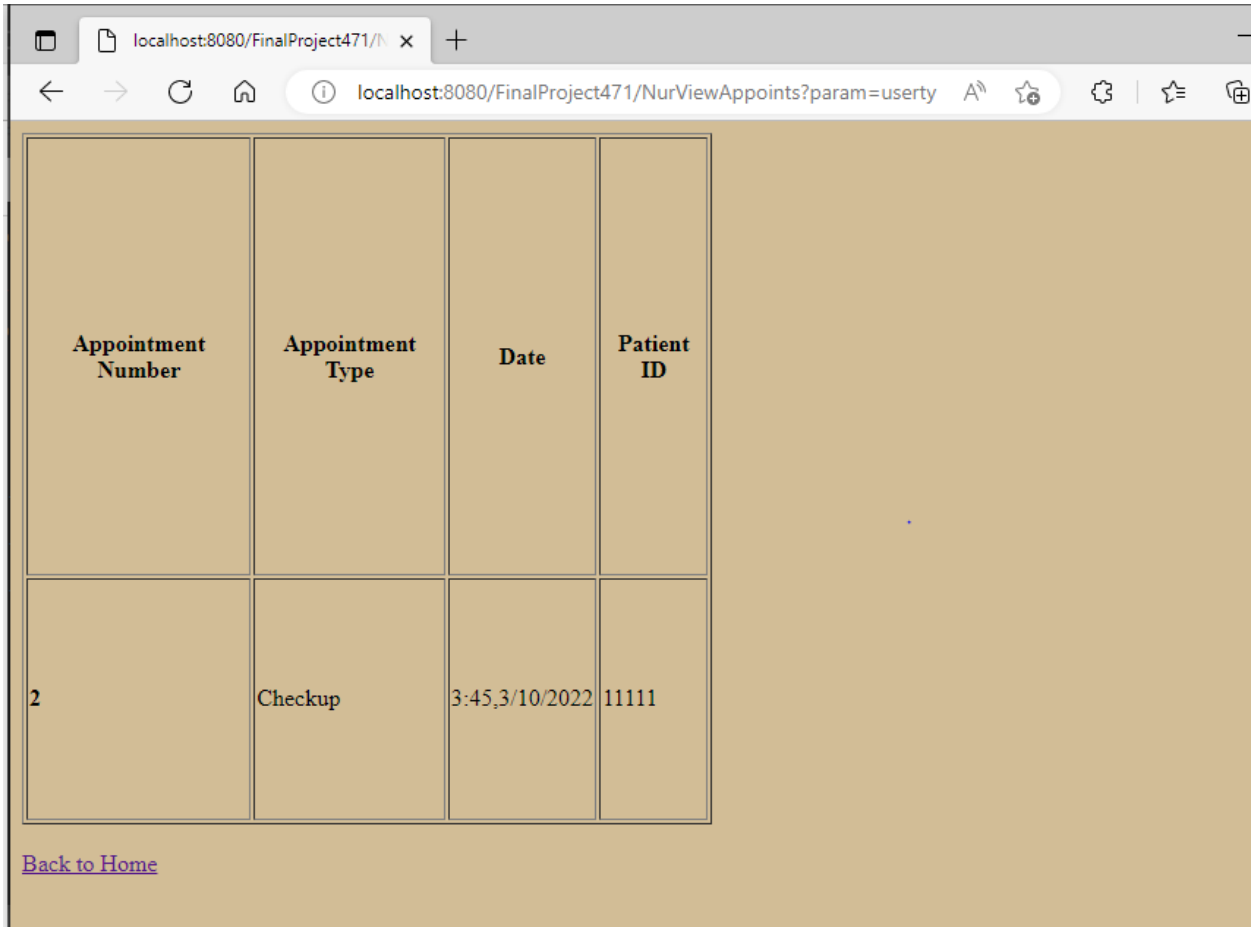
When a patient books an appointment, a nurse that works at the clinic the patient is booking at is randomly selected to be the nurse of the appointment. As such, since no patient has made an appointment yet since we registered and created this nurse there will be no appointments yet for this nurse:



View My Appointments for this nurse, who has no appointments yet.

To show this, lets quickly logout and login to a pre-existing nurse who has an appointment. If we look at Appendix U.21 we can see that Nurse ID=123 and Nurse Username=userty has a pre-existing appointment booked. As such, we will login with this user and select view your appointments simply to view this option's functionality when their appointment exists.

When we do that, we get this:

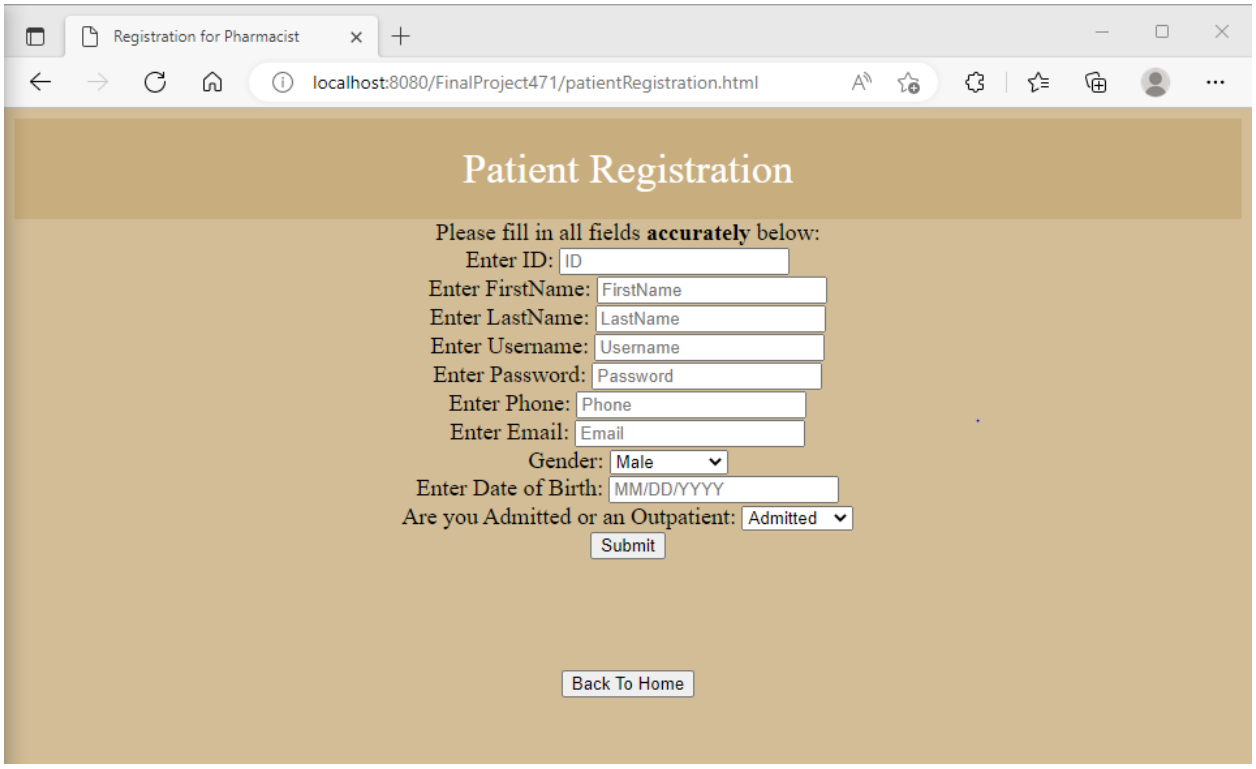


View Your Appointment for Nurse Jaa Daniels, which is the user seen in appendix u21 who has an appointment

Now we can logout from this user, and we are finished with everything a nurse can do in our system. Now we can logout and move on to talking about the patient user's capabilities.

Patient Functionalities

Now we'll discuss patient. First, lets look at the patient registration:

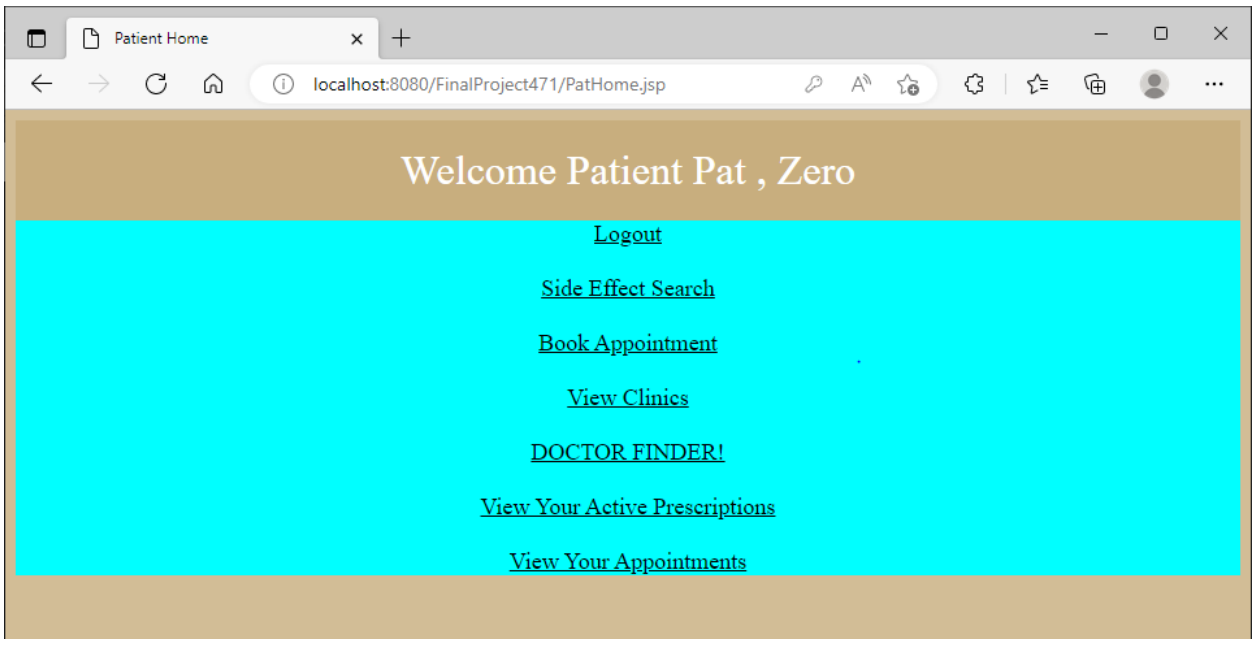


The patient registration page and form

Here, we can fill in the data to create a new patient user.

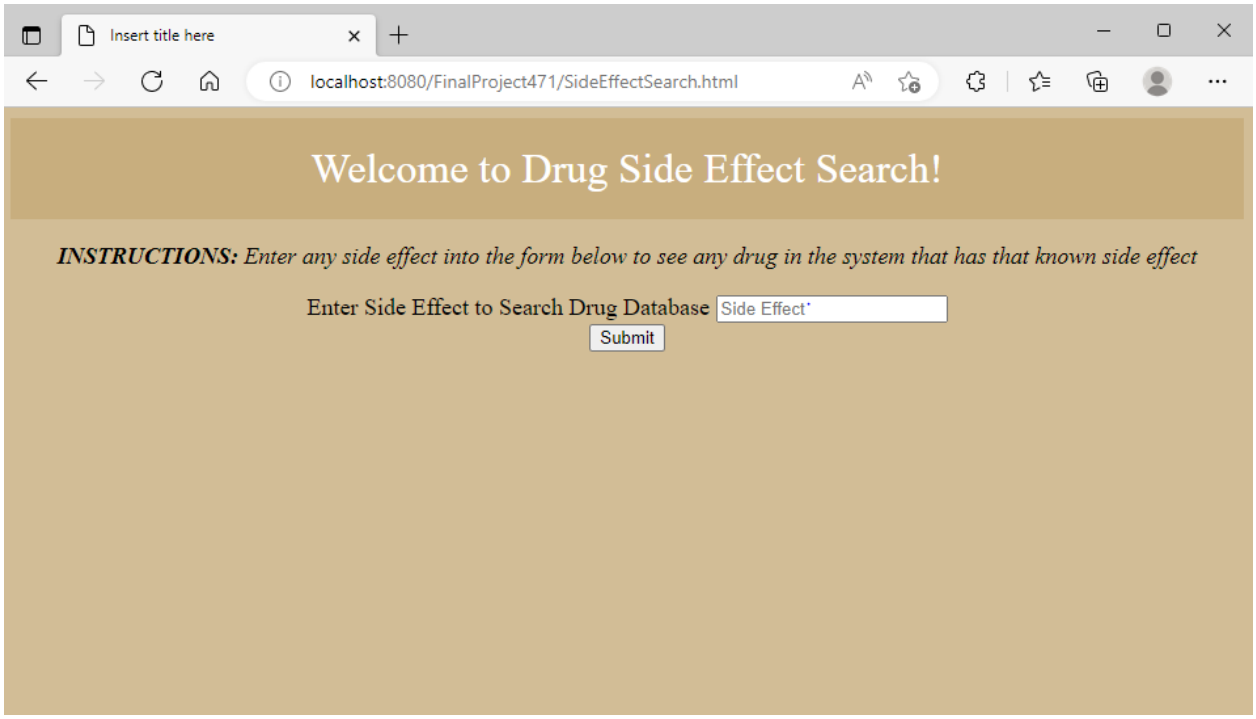
The login functionality for the patient is the exact same as the other users so we can skip that for brevity. Refer to any of the above users to see the login functionality of our system.

Now we can login to an existing user: Pat Zero.



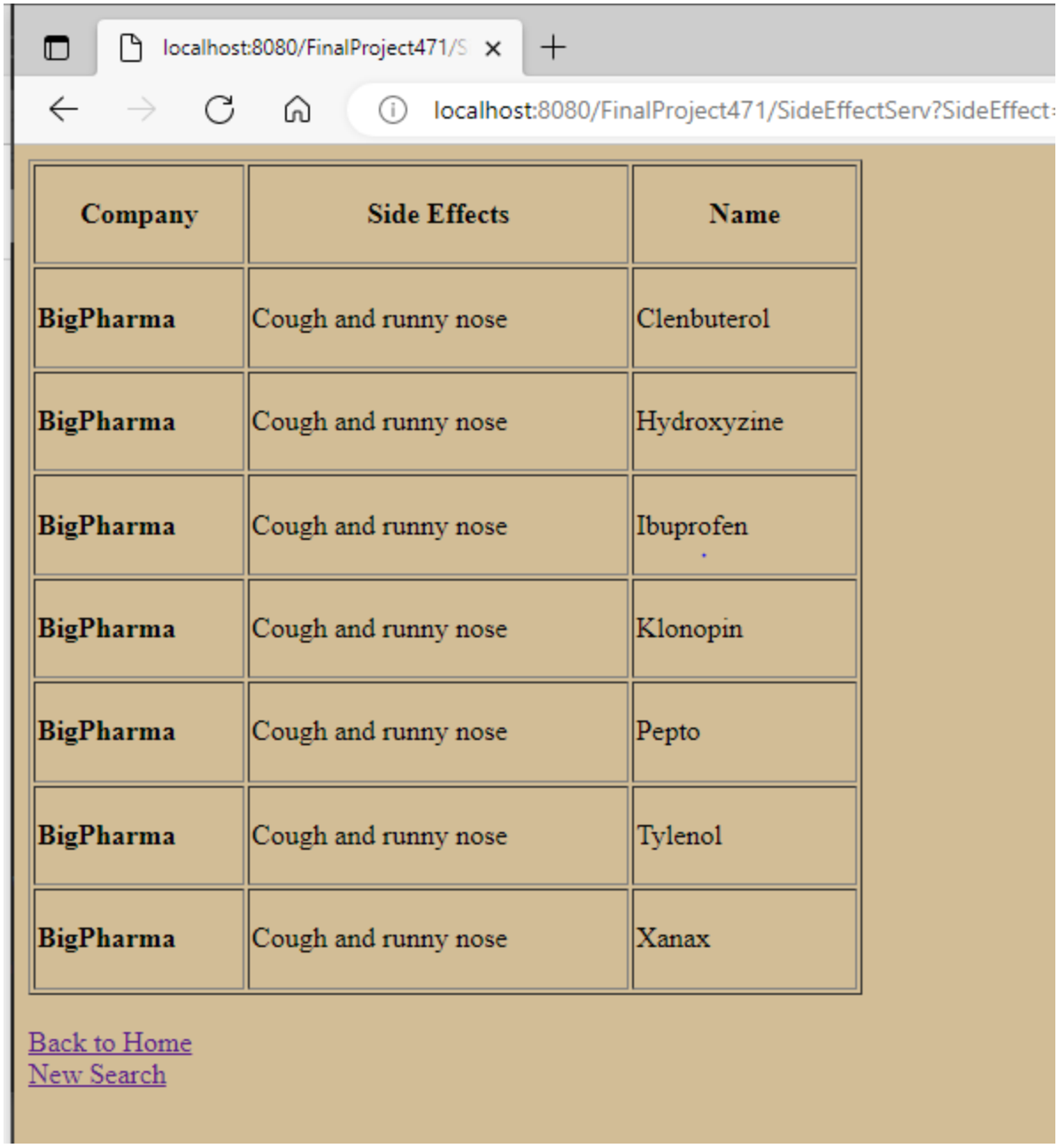
Patient Homepage after logging in as Pat Zero.

Now we are at the patient homepage. First, we will examine the first option: Side Effect Search.

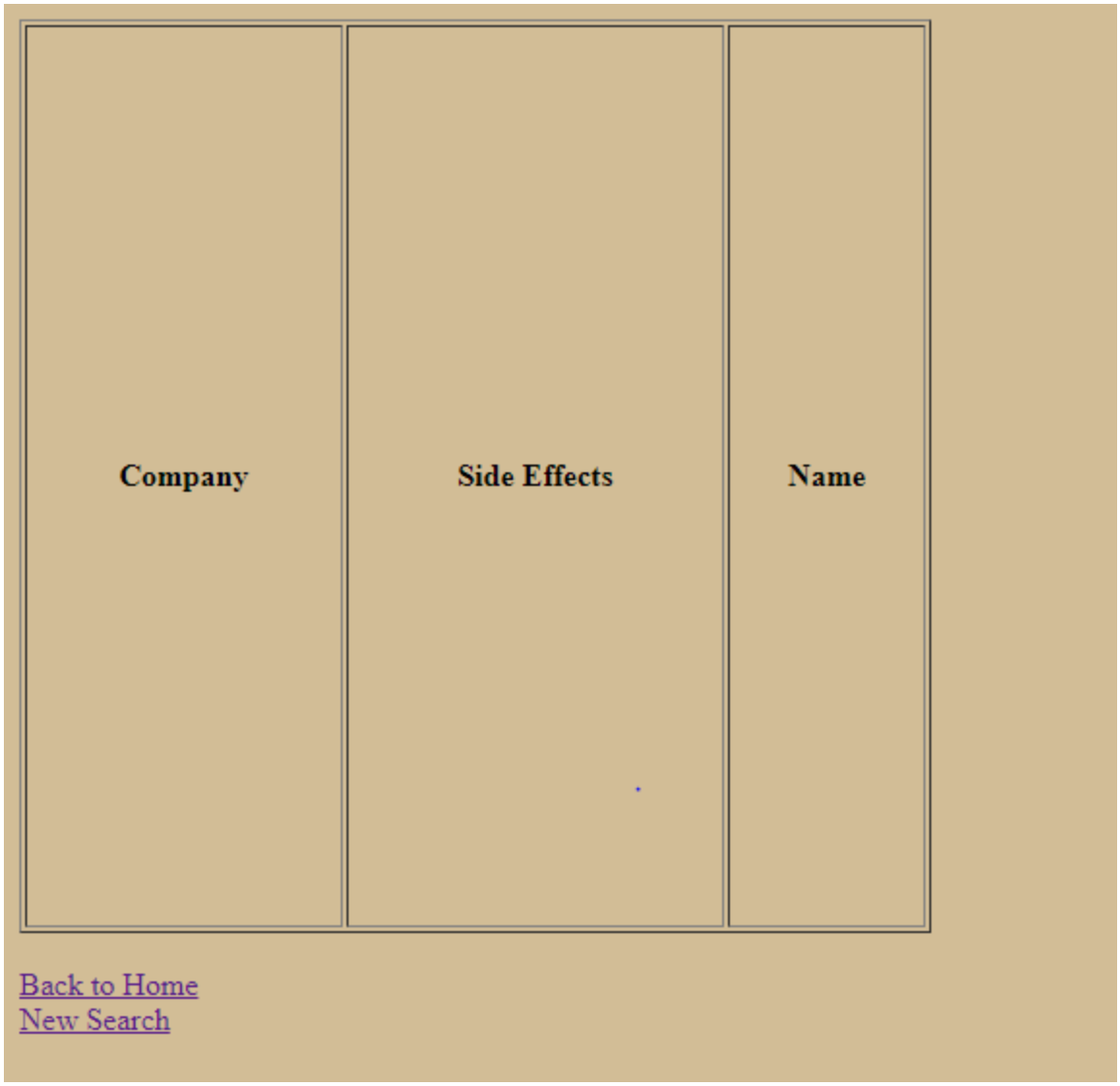


Patient Side Effect Search

Now we're brought to the side effect search page, which gives us a text entry to search a side effect, and it will retrieve any drugs that have that may have that side effect in its list.
See appendix u.1 for drugs to verify drug's side effects.

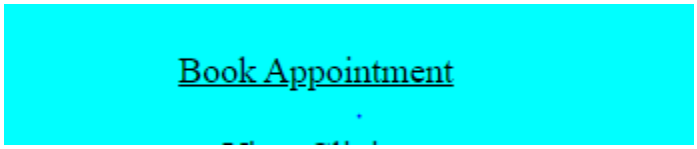


Side effect search for 'cough'



Side effect search for a side effect that never appears in database drugs

Now we can go back to the patient homepage and select the next patient option: book appointment.

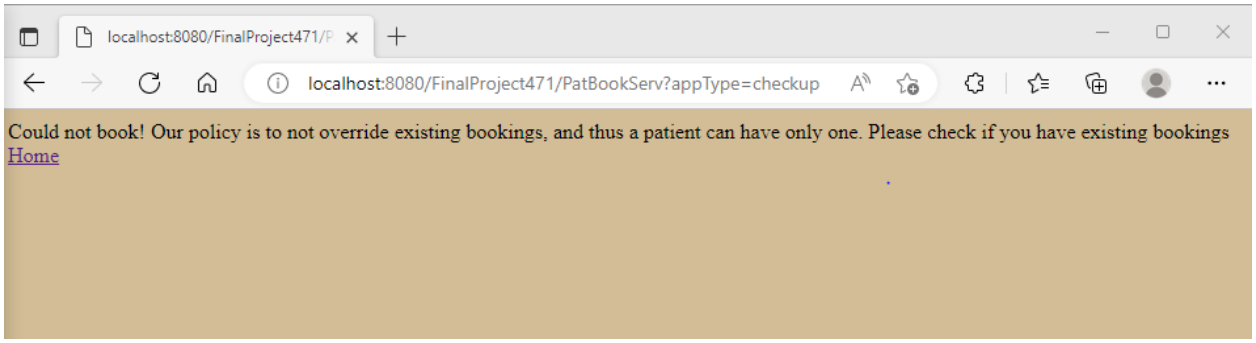


Book appointment

The first step in booking is entering the appointment type.

Step one of booking appointment

If we enter ‘checkup’ into the text field and submit, we get:

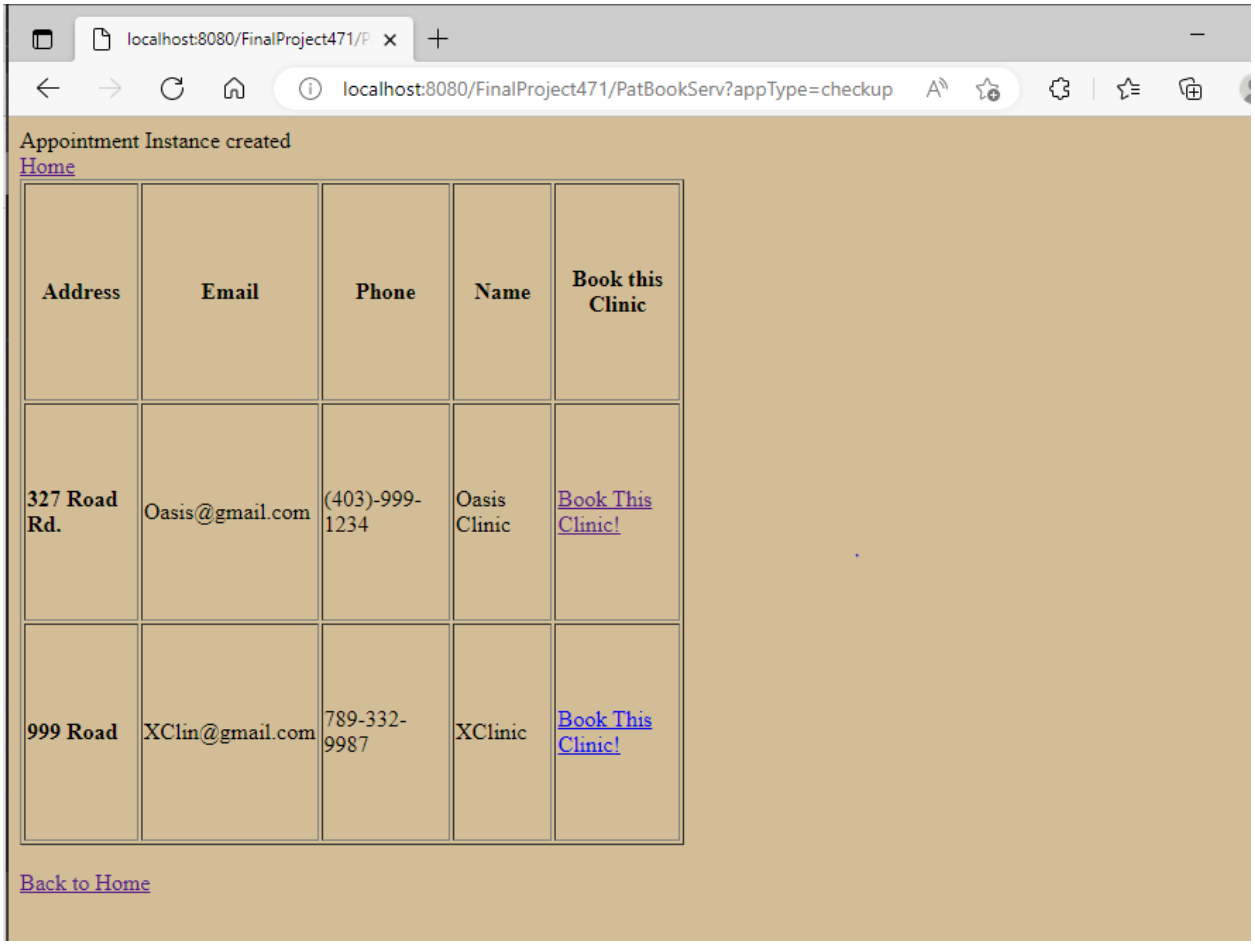


Message not booked message

The policy of our system is to have only one active booking. According to appendix U22, which shows that this logged in user indeed has an active booking, and this is why they cannot overwrite their booking.
In order to show booking an appointment fully, we will logout and login to Frank Ocean’s patient account (who does not have a booking) to fully show a valid book.

Now we’ll login to Frank Ocean’s account, select book an appointment, enter checkup for appointment type.

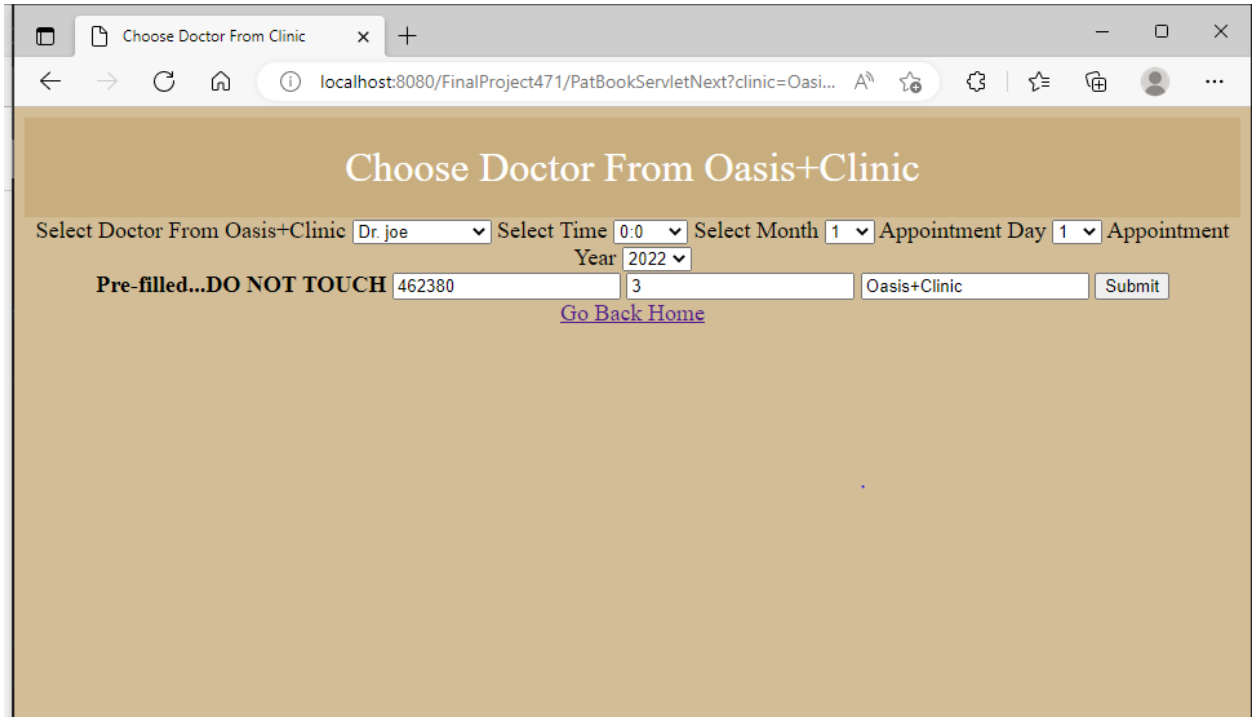
Now we see this:



Step after entering appointment type: Booking clinic

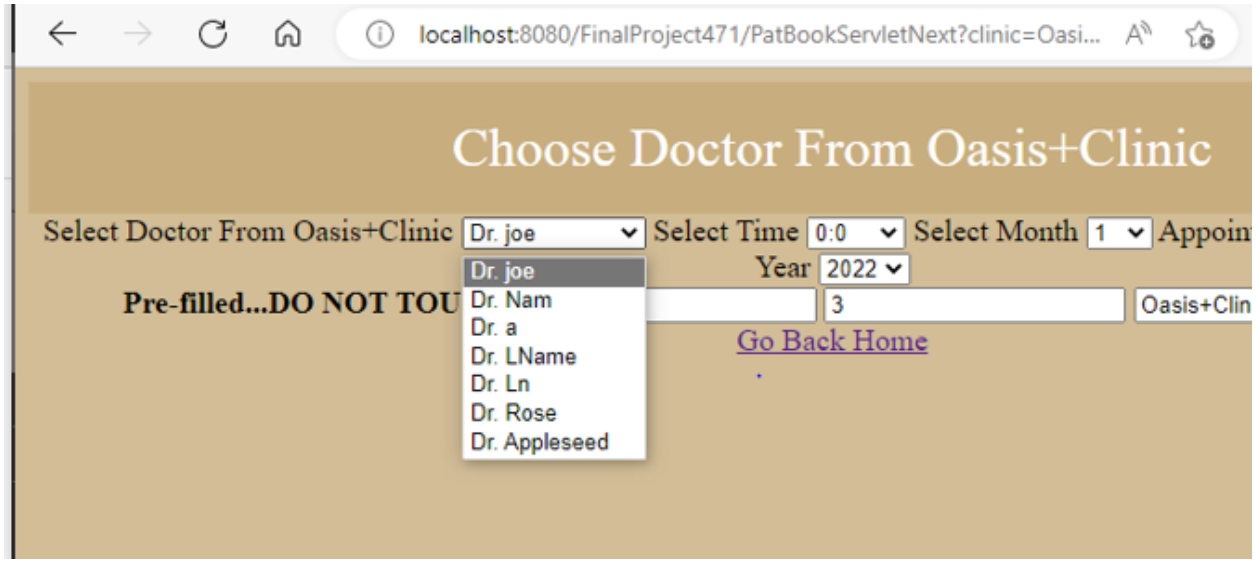
Now we are prompted to book a clinic. Every existing clinic in the database and its info is printed, with a link to book this clinic. Pressing book this clinic! for a row will set the booking clinic to that row's clinic.

For our example, we will select Oasis Clinic and we will then see:



Next step in booking: selecting doctor

Now we have the option to select the doctor from the clinic we selected. The dropdown menu for doctor contains every doctor that is at that selected clinic only:



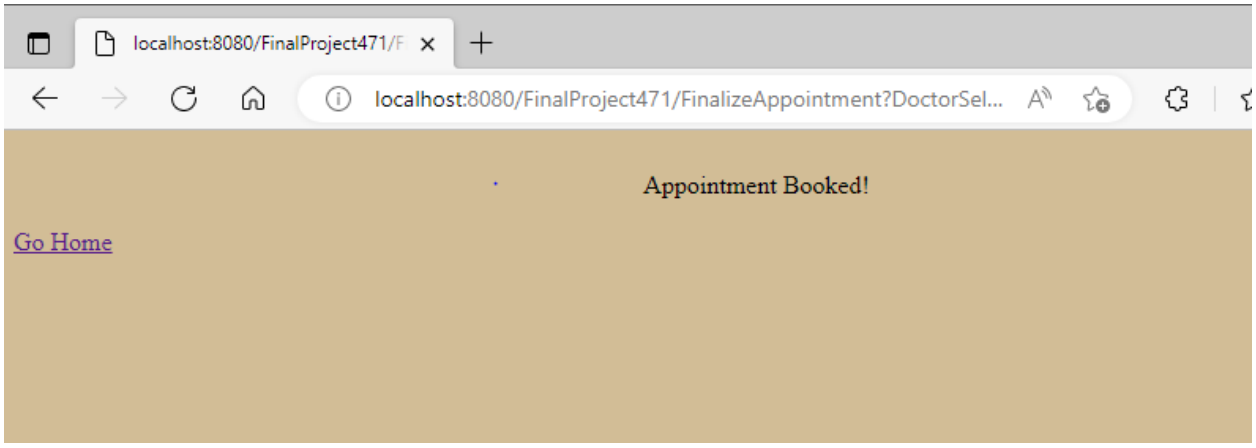
All doctors who are in Oasis Clinic (selected clinic). View appendix u23 to see this further.

Similarly, the time has all times from 0:0 to 23:45 with 15 minute intervals, month has 1-12, day has 1-31. The last text fields are pre-determined from previous steps and should not be altered.

Lets fill out the booking as this:

Sample form fill in to book an appointment with Dr.Appleseed at Oasis Clinic

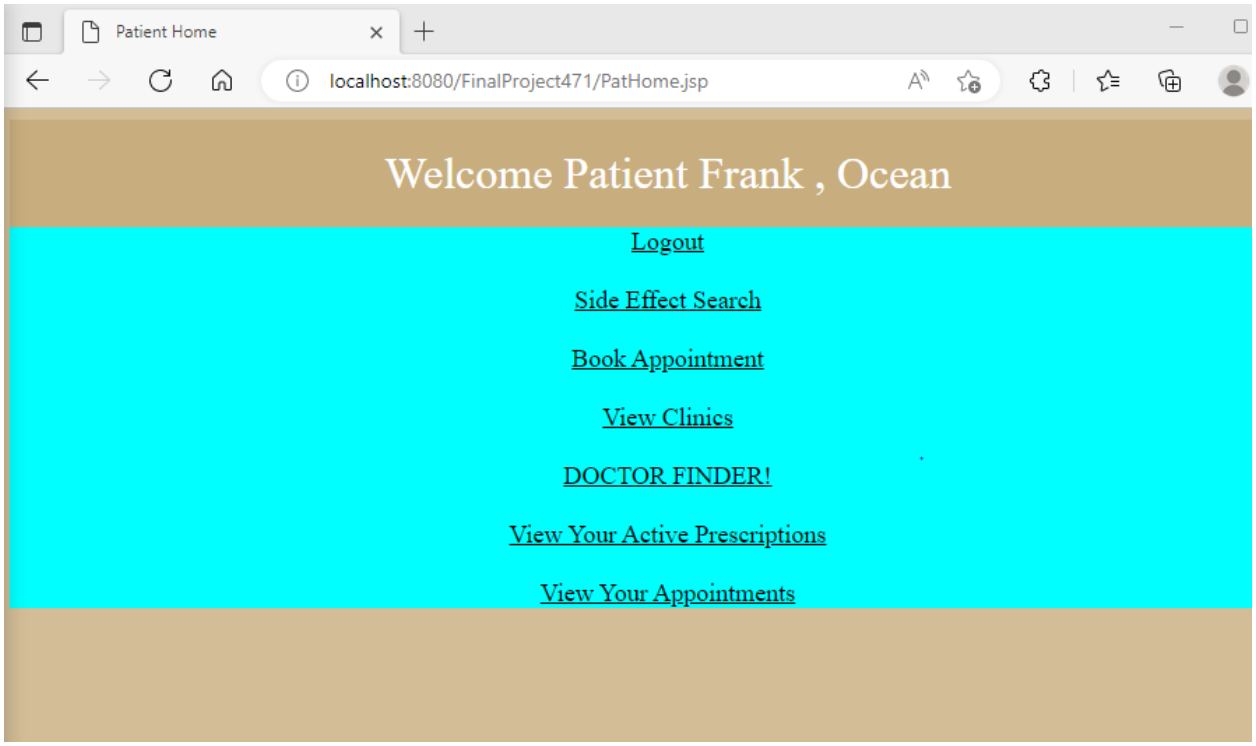
Now we get this message:



Successful booking

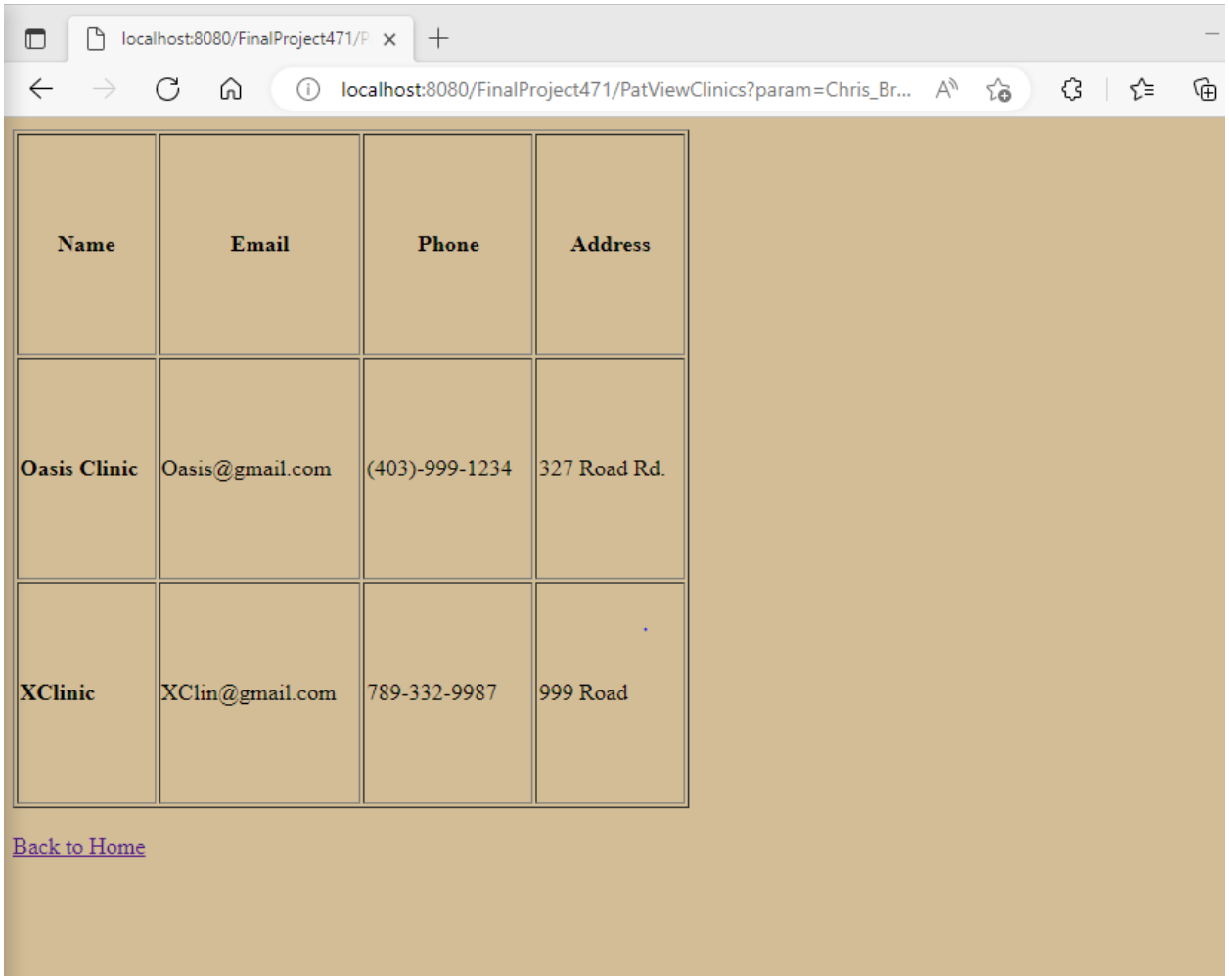
See appendix u24 to see this addition to the database system.
Notice that the system gives a calculated unique appointment number to the appointment instance, and the nurse is auto assigned by randomly assigning one nurse who works at that specific clinic to be the nurse. Comparing appendix u24 to u22 shows the before and after for our database.

Now we can move on to the next patient ability: View Clinics.



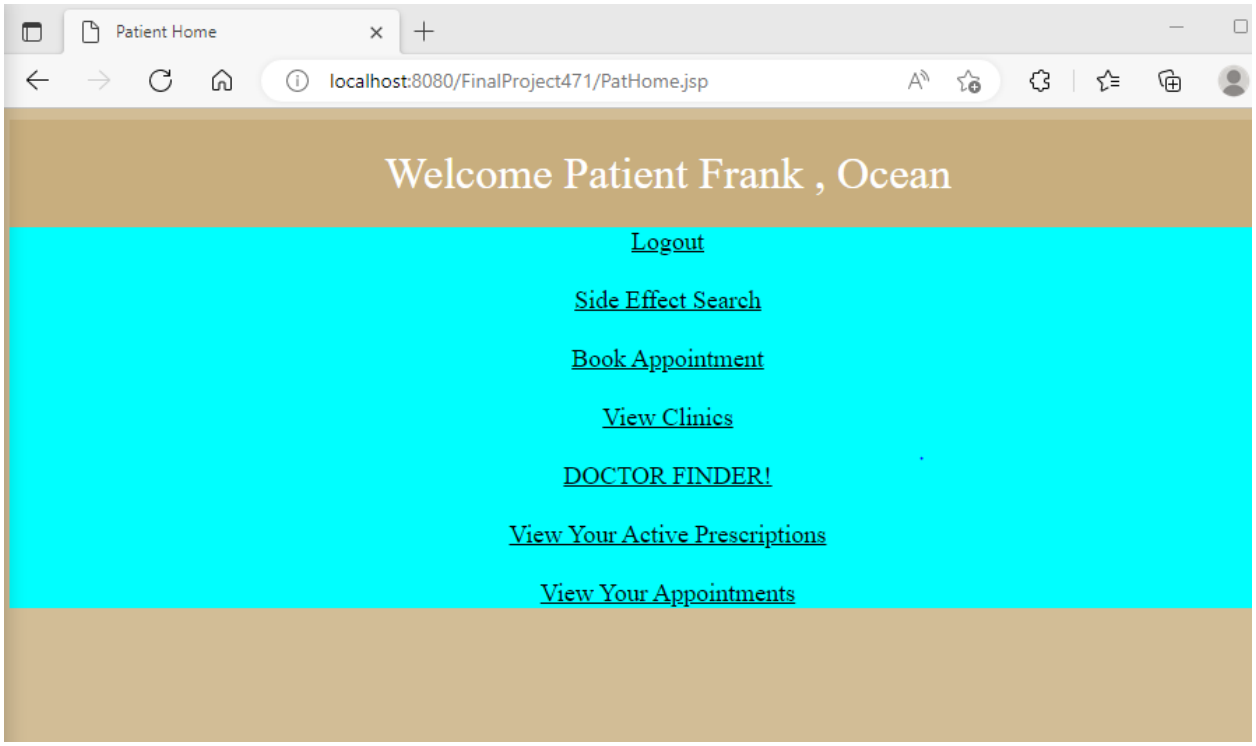
Patient Homepage. View Clinics is the fourth option

After selecting View Clinics, we see the system retrieve info for every clinic in the system:



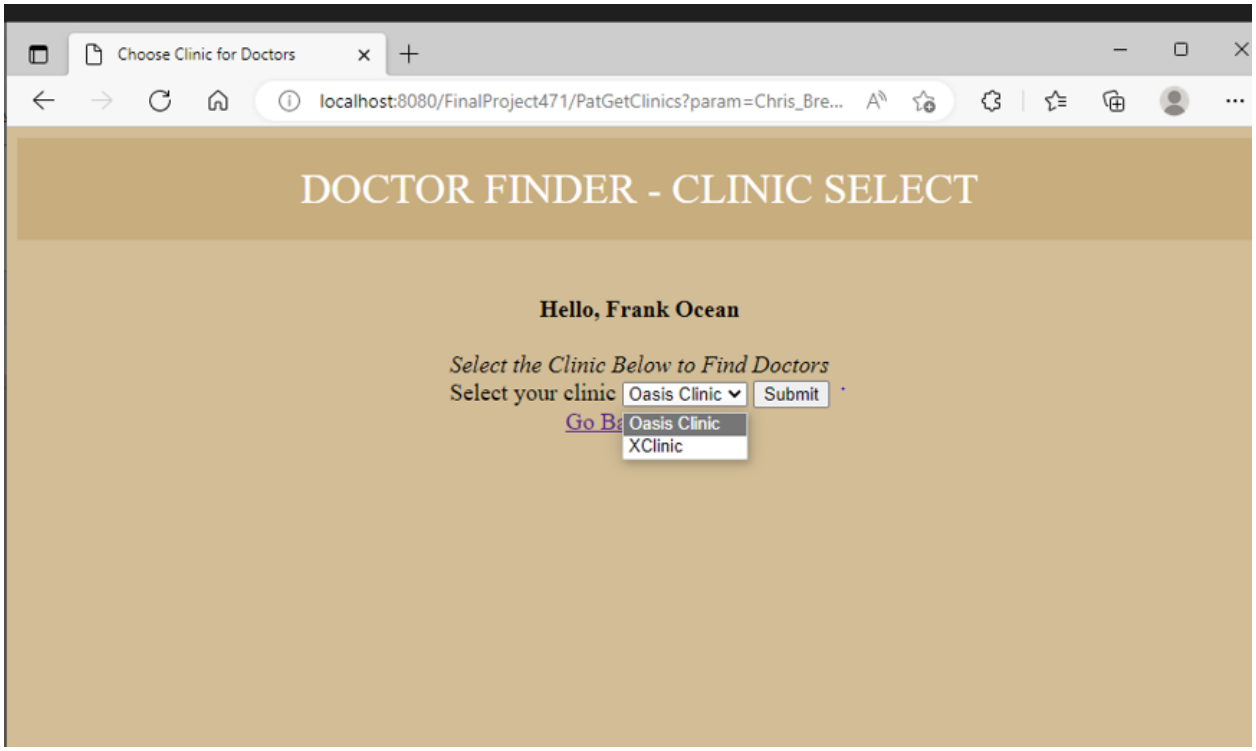
Patient View Clinics result

The next functionality to show is the Doctor Finder (5th option).



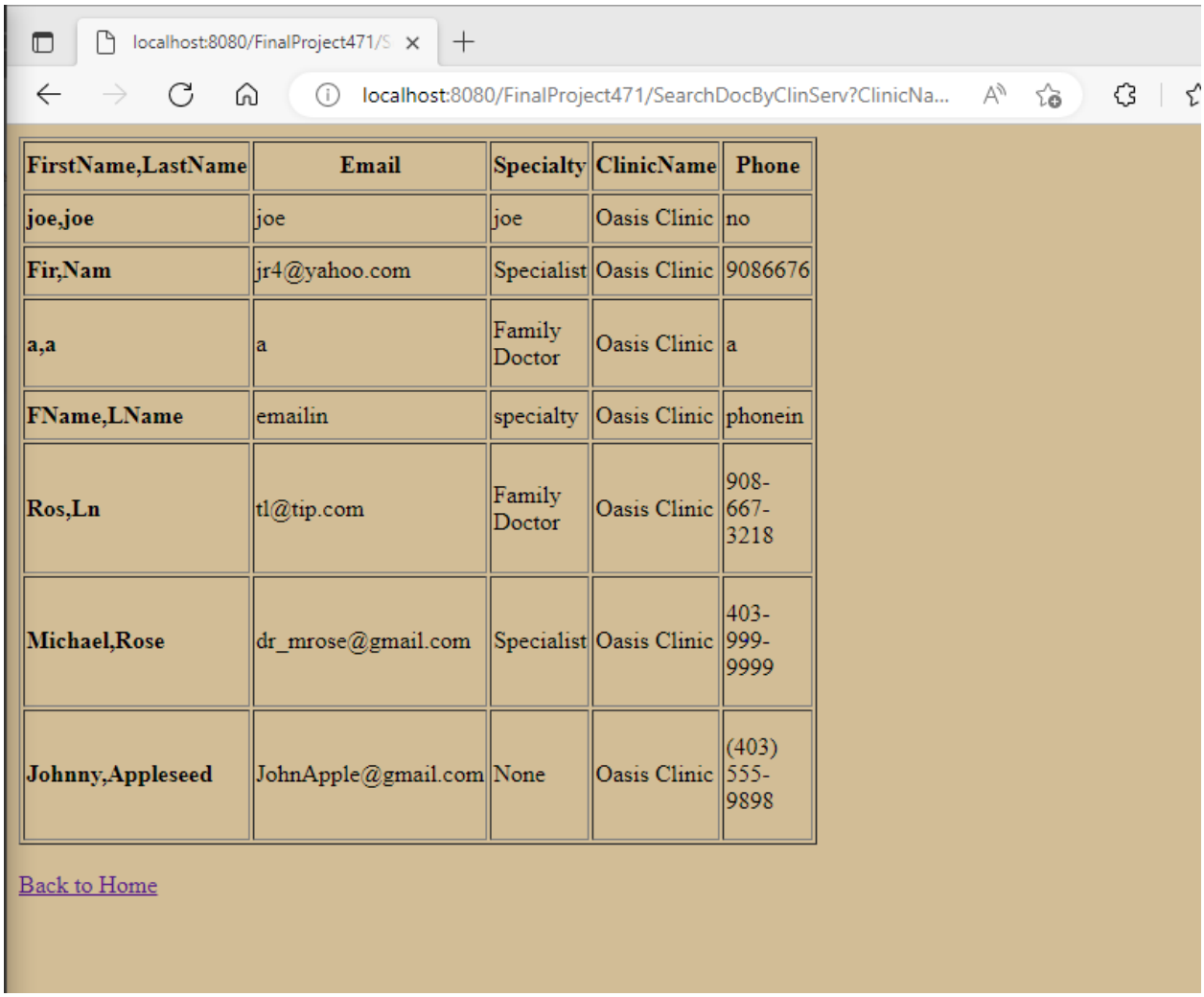
Patient homepage. Doctor Finder is 3rd from the bottom.

Selecting doctor finder gives us this page:



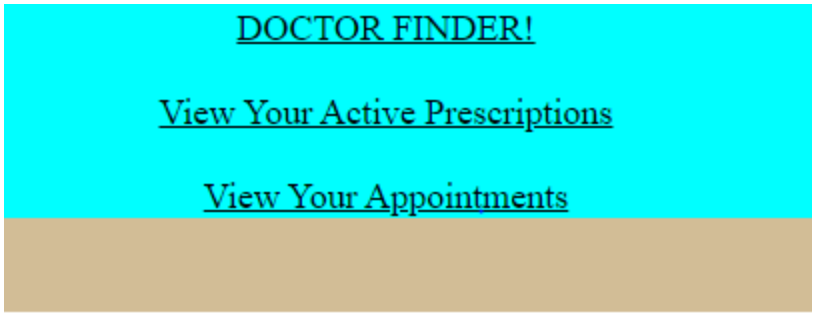
Doctor finder first step - selecting the clinic

As you can see, you have the option to select the clinic you wish to see doctors for based on the dropdown menu.
Now if we select Oasis Clinic we see all doctors who work at oasis clinic (matches appendix u23), and some selected information for that doctor.



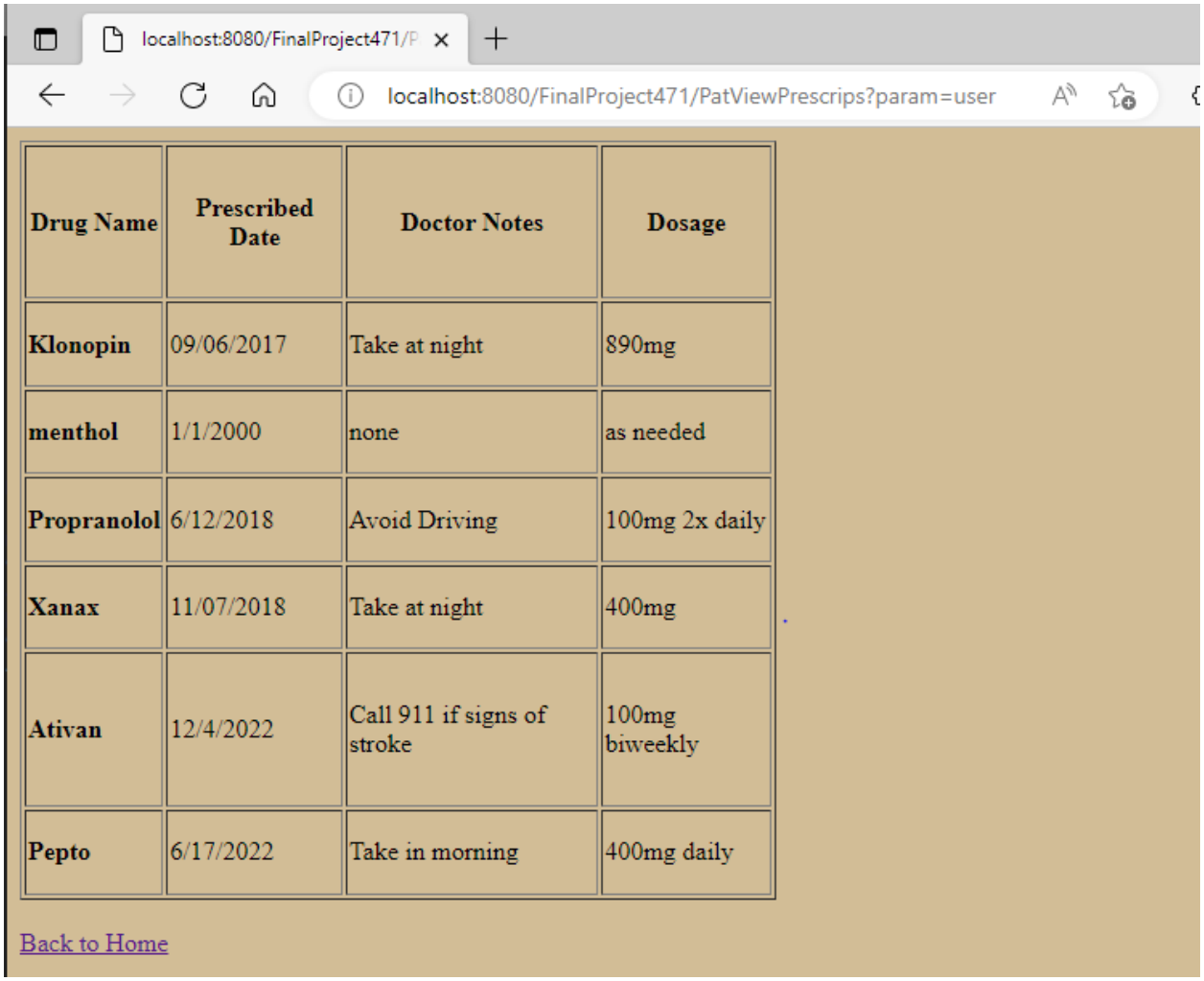
Selecting Oasis Clinic for doctor search and submitting

Now we can go back to the patient's homepage and look at the final two patient options: view active prescriptions and view your appointments.



The bottom of patient's homepage with the final two options underneath doctor finder

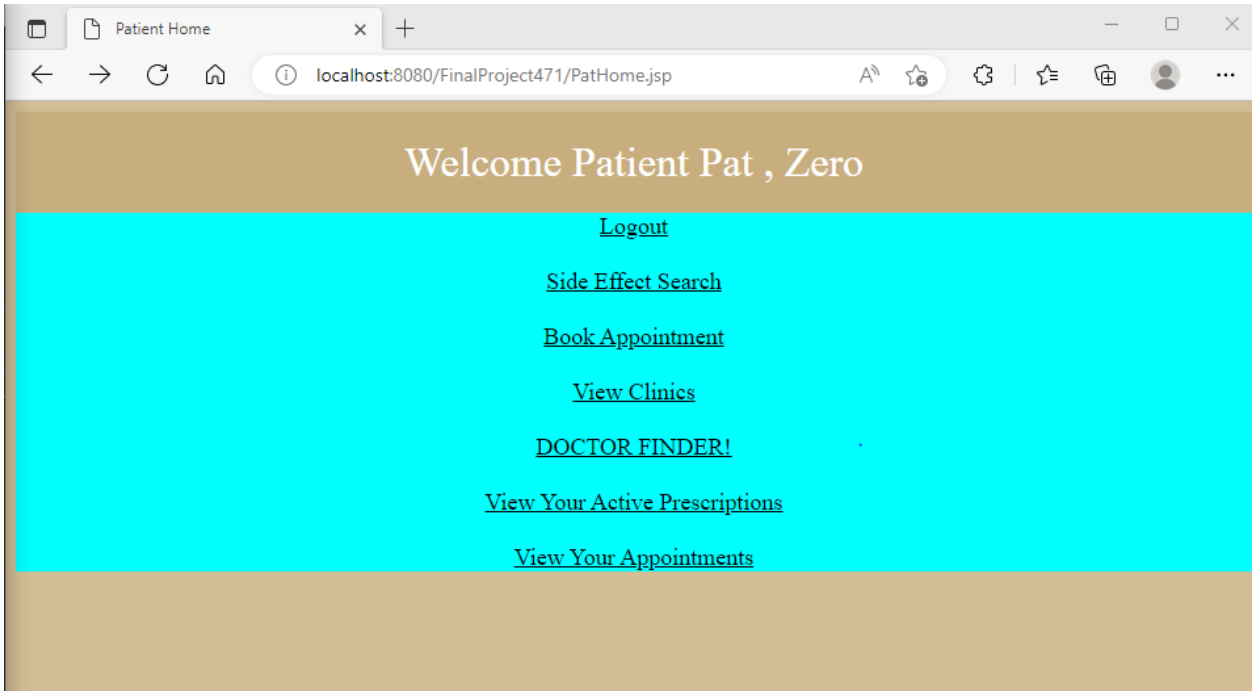
First, we select View Active Prescriptions. Recall that this is connected to prescriptions prescribed by the doctor earlier.
Since this user has no active prescriptions (and thus an empty table), we will login to Pat Zero's account to demonstrate this function.
Once we select view active prescriptions in this account, we see the following:



View Active Prescriptions for patient Pat Zero

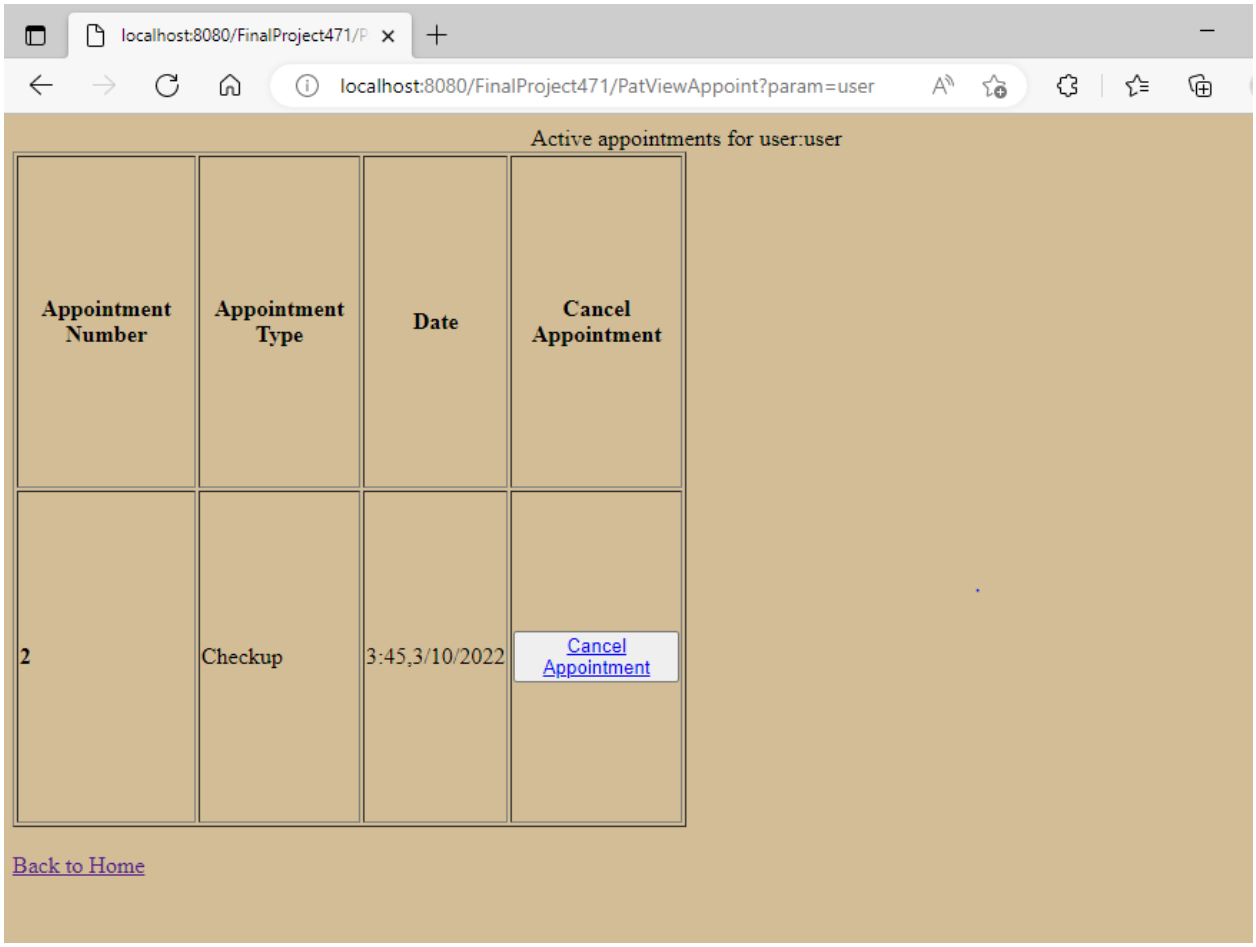
It is evident that here the patient can view all prescriptions prescribed to them from doctors, including all notes and information about those drugs. Note this correlates to the prescribed table in appendix u24.

Now, using Pat Zero's account still, we can show the last option on the patient homepage: View Your Appointments:



Patient homepage - view appointments is the final option

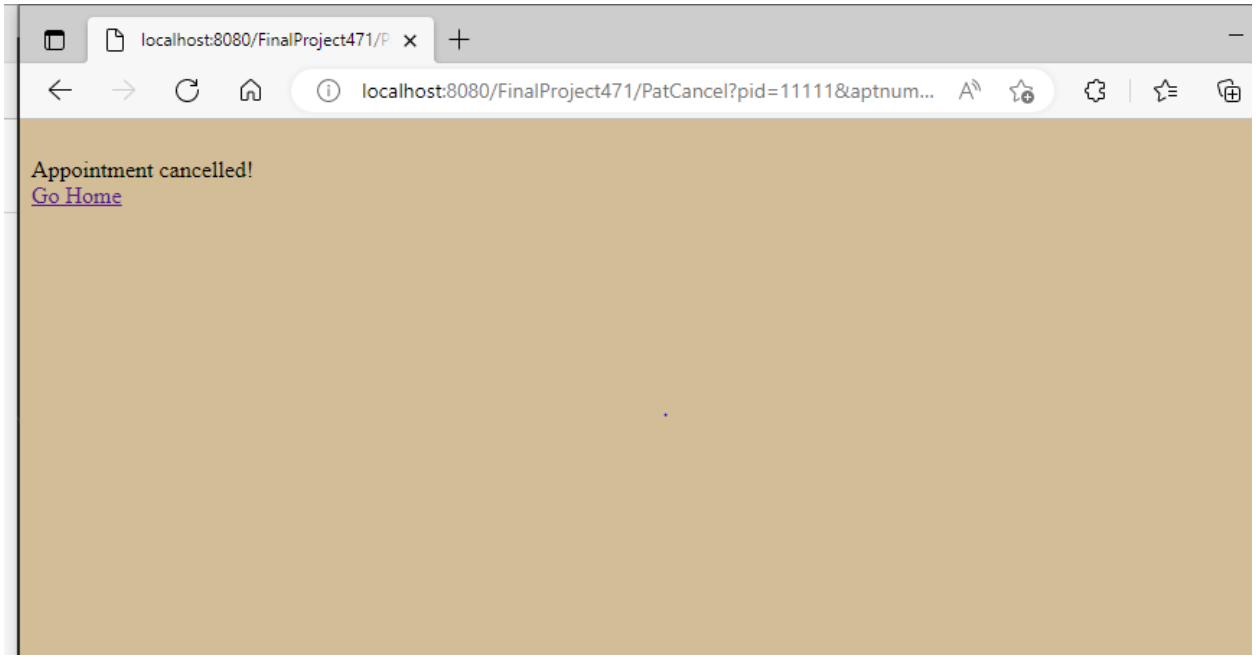
Selecting this option gives us this page:



Result of selecting view appointments for Patient Pat Zero

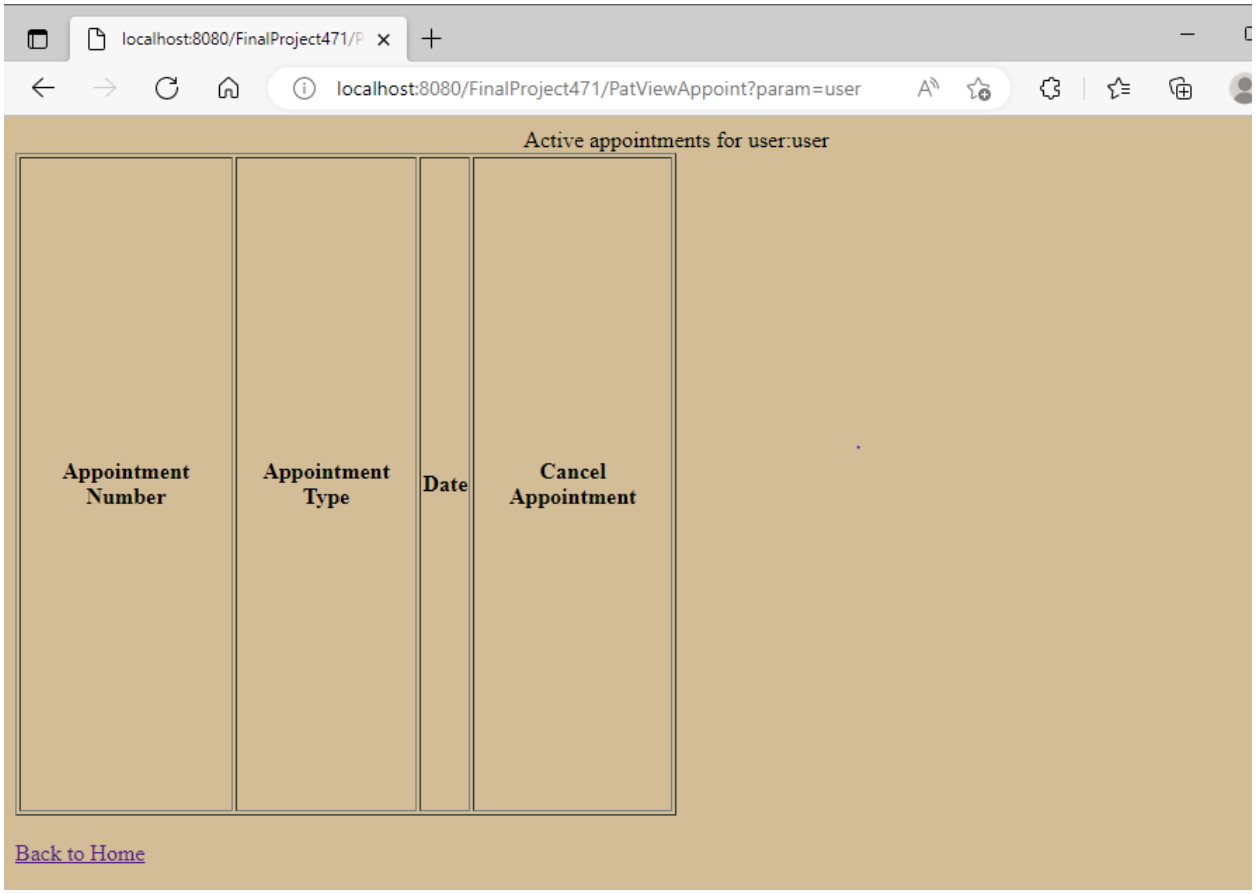
Here we see the information for our booked appointment for this patient, and have the option to cancel the appointment via the cancel appointment button seen above.

If we press this, we see:



Successfully cancelled appointment for Pat Zero

See appendix u26 to see the changes (compared to u25). Now if we go back and view appointments, we see:



No appointments now for Pat Zero following cancellation

Now this patient can go and book a new appointment (by selecting book appointment on their homepage), since they do not have an active appointment that would need to be cancelled first.

That concludes our discussion for patient.

That concludes the user manual, which has provided a thorough description of every possibility of our created webpage for every single user, as well as tracking the transactional changes to the underlying database via the appendix (u1-u26). This user manual has thus fully described the entire functionality and the transactions for our created website.

Appendix

U.1: Database State Before User Manual(multiple images):

```
mysql> use medicalsystem;
Database changed
mysql> show tables;
+--------------------------+
| Tables_in_medicalsystem  |
+--------------------------+
| appointment              |
| clinic                   |
| contains                 |
| doctor                   |
| doctor_address           |
| doctorschedule           |
| drugs                    |
| has                      |
| nurse                    |
| nurseschedule            |
| patient                  |
| patient_address          |
| patient_conditions       |
| patient_medicalhis       |
| pharmacist               |
| prescribes               |
| receives                 |
+--------------------------+
17 rows in set (0.04 sec)
```

U.1.1: All tables in database

```
mysql> select * from doctor;
+-------+---------+-------+--------------------+--------------+-------------+-----------+----------+----------+--------------+------------+
| ID    | Fname   | Lname | Email              | Specialty    | ClinicName  | Username  | Spe_Flag | Fam_Flag | Phone        | Password   |
+-------+---------+-------+--------------------+--------------+-------------+-----------+----------+----------+--------------+------------+
|    12 | joe     | joe   | joe                | joe          | Oasis Clinic | koe       |        1 |        0 | no           | nope       |
|    32 | Fir     | Nam   | jr4@yahoo.com      | Specialist   | Oasis Clinic | fna@      |        1 |        0 | 9086676      | 12345tyu   |
|   123 | a       | a     | a                  | Family Doctor | Oasis Clinic | a         |        1 |        0 | a            | a          |
|   123 | FName   | LName | emailin            | specialty    | Oasis Clinic | userin    |        1 |        0 | phonein      | pswdin     |
|   453 | Ros     | Ln    | tl@tip.com         | Family Doctor | Oasis Clinic | r_ln      |        0 |        1 | 908-667-3218 | pass       |
| 89765 | Michael | Rose  | dr_mrose@gmail.com | Specialist   | Oasis Clinic | mrose4567 |        1 |        0 | 403-999-9999 | iloveroses |
+-------+---------+-------+--------------------+--------------+-------------+-----------+----------+----------+--------------+------------+
6 rows in set (0.00 sec)
```

U.1.2: All doctors

```
mysql> select * from nurse;
+-----+-------+---------+--------------+---------------+----------+------------+
| ID  | Fname | Lname   | ClinicName   | Specialty     | Username | Password   |
+-----+-------+---------+--------------+---------------+----------+------------+
|   3 | Ri    | Lu      | Oasis Clinic | Many of them  | pas      | word       |
| 123 | Jaa   | Daniels | Oasis Clinic | Special Nurse | userty   | passwordty |
+-----+-------+---------+--------------+---------------+----------+------------+
2 rows in set (0.00 sec)
```

U.1.3: All Nurses

```
mysql> select * from pharmacist;
+-------+--------+-------+-----------+------------+----------+
| ID    | Fname  | Lname | Supply    | Username   | Password |
+-------+--------+-------+-----------+------------+----------+
|    56 | Jordan | Swiss | none      | cat_nyr    | eric     |
|  1234 | John   | Doe   | SupplyOne | JohnDoe123 | mypass   |
| 12345 | jar    | lan   | SupplyO i | jr_l       | 123ref   |
+-------+--------+-------+-----------+------------+----------+
3 rows in set (0.01 sec)
```

U.1.4: All pharmacists

```
mysql> select * from clinic;
+--------------+-----------------+----------------+--------------+
| Address      | Email           | Phone          | Name         |
+--------------+-----------------+----------------+--------------+
| 327 Road Rd. | Oasis@gmail.com | (403)-999-1234 | Oasis Clinic |
| 999 Road     | XClin@gmail.com | 789-332-9987   | XClinic      |
+--------------+-----------------+----------------+--------------+
2 rows in set (0.01 sec)
```

U.1.5: All clinics

```
mysql> select * from appointment;
+-------+-------------------+-----------------+
| ID    | AppointmentNumber | AppointmentType |
+-------+-------------------+-----------------+
| 11111 |                 2 | Checkup         |
+-------+-------------------+-----------------+
1 row in set (0.00 sec)
```

U.1.6: All appointments

```
mysql> select * from doctorschedule;
+-----+-------+-------------+----------------+----------+
| ID  | Hours | VacatonDays | Days           | Username |
+-----+-------+-------------+----------------+----------+
|  12 |     8 | Mondays     | T-F            | koe      |
| 123 |    18 | july 22     | Monday-Tuesday | userin   |
+-----+-------+-------------+----------------+----------+
2 rows in set (0.01 sec)
```

U.1.7: All doctorschedule

```
mysql> select * from drugs;
+--------------+---------------------------------------+-------------+
| Company      | SideEffects                           | Name        |
+--------------+---------------------------------------+-------------+
| 123          | 123                                   | 123         |
| Balmoral     | None                                  | Ativan      |
| BigPharma    | Cough and runny nose                  | Clenbuterol |
| NULL         | NULL                                  | Clorazepam  |
| BigPharma    | Cough and runny nose                  | Hydroxyzine |
| BigPharma    | Cough and runny nose                  | Ibuprofen   |
| BigPharma    | Cough and runny nose                  | Klonopin    |
| naturalPharm | none                                  | menthol     |
| PharmMD      | Dizziness, diarrhea, confusion, etc.  | Metranolol  |
| BigPharma    | Cough and runny nose                  | Pepto       |
| Drug Inc.    | Drowsiness, Irritability              | Propranolol |
| BigPharma    | Cough and runny nose                  | Tylenol     |
| BigPharma    | Cough and runny nose                  | Xanax       |
+--------------+---------------------------------------+-------------+
13 rows in set (0.01 sec)
```

U.1.8: All Drugs

```
mysql> select * from has;
+----------+---------+-----------+-------------------+-------------+---------+-----------+----------------+
| DoctorID | NurseID | PatientID | AppointmentNumber | PatientUser | DocUser | NurseUser | TimeDate       |
+----------+---------+-----------+-------------------+-------------+---------+-----------+----------------+
|       32 |     123 |     11111 |                 2 | user        | fna@    | userty    | 3:45,3/10/2022 |
+----------+---------+-----------+-------------------+-------------+---------+-----------+----------------+
1 row in set (0.01 sec)
```

U.1.9: All has (corresponds with appointment)

```
mysql> select * from nurseschedule;
+----+-------+-------------+-------------------+----------+
| ID | Hours | VacatonDays | Days              | Username |
+----+-------+-------------+-------------------+----------+
|  3 |    10 | may 12      | Tuesday-Saturday  | pas      |
+----+-------+-------------+-------------------+----------+
1 row in set (0.01 sec)
```

U.1.10: All nurse schedule

```
mysql> select * from patient;
+--------+-------+-------+-----------+------------+------------+----------+----------+--------------+---------------+
| ID     | Fname | Lname | Email     | DoB        | Gender     | Adm_Flag | Out_Flag | Username     | Password      |
+--------+-------+-------+-----------+------------+------------+----------+----------+--------------+---------------+
|  11111 | Pat   | Zero  | Jar@33e   | 04/23/1999 | Non-Binary |        0 |        1 | user         | pass          |
| 462380 | Frank | Ocean | jjjjjjjjj | 04/13/5555 | Male       |        0 |        1 | Chris_Breaux | ChannelOrange |
+--------+-------+-------+-----------+------------+------------+----------+----------+--------------+---------------+
2 rows in set (0.01 sec)
```

U.1.11: All patient

```
mysql> select * from patient_conditions;
+-------+-----------------------------------------------------------------+
| ID    | Conditions                                                      |
+-------+-----------------------------------------------------------------+
| 11111 | Routine Spinal Tap (2009) , Aertrial Bypass Surgery(08/2008) |
+-------+-----------------------------------------------------------------+
1 row in set (0.00 sec)
```

U.1.12: All patient_conditions

```
mysql> select * from patient_medicalhis;
+-------+-------------------------+
| ID    | MedicalHistory          |
+-------+-------------------------+
| 11111 | Heart AttackBad Cough |
+-------+-------------------------+
1 row in set (0.01 sec)
```

U.1.13: All patient_medicalhis

```
mysql> select * from prescribes;
+-----+-------------+----------+------------+-----------+---------------+-----------------+
| ID  | Name        | Username | Prescribed | PatientID | DoctorNotes   | Dosage          |
+-----+-------------+----------+------------+-----------+---------------+-----------------+
| 123 | Klonopin    | a        | 09/06/2017 | 11111     | Take at night | 890mg           |
| 123 | menthol     | a        | 1/1/2000   | 11111     | none          | as needed       |
| 123 | Propranolol | a        | 6/12/2018  | 11111     | Avoid Driving | 100mg 2x daily |
| 123 | Xanax       | a        | 11/07/2018 | 11111     | Take at night | 400mg           |
+-----+-------------+----------+------------+-----------+---------------+-----------------+
4 rows in set (0.01 sec)
```

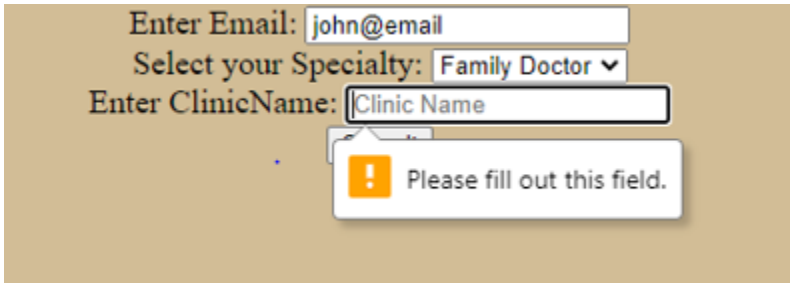
U.1.14: All prescribes (related to drugs, etc.)

U.2: Doctors after registering ‘Johnny Appleseed’

```
mysql> select * from doctor;
+--------+---------+----------+---------------------+--------------+-------------+----------+----------+----------+----------------+-----------+
| ID     | Fname   | Lname    | Email               | Specialty    | ClinicName  | Username | Spe_Flag | Fam_Flag | Phone          | Password  |
+--------+---------+----------+---------------------+--------------+-------------+----------+----------+----------+----------------+-----------+
|     12 | joe     | joe      | joe                 | joe          | Oasis Clinic | koe      |        1 |        0 | no             | nope      |
|     32 | Fir     | Nam      | jr4@yahoo.com       | Specialist   | Oasis Clinic | fna@     |        1 |        0 | 9086676        | 12345tyu  |
|    123 | a       | a        | a                   | Family Doctor | Oasis Clinic | a        |        1 |        0 | a              | a         |
|    123 | FName   | LName    | emailin             | specialty    | Oasis Clinic | userin   |        1 |        0 | phonein        | pswdin    |
|    453 | Ros     | Ln       | tl@tip.com          | Family Doctor | Oasis Clinic | r_ln     |        0 |        1 | 908-667-3218   | pass      |
|  89765 | Michael | Rose     | dr_mrose@gmail.com  | Specialist   | Oasis Clinic | mrose4567 |        1 |        0 | 403-999-9999   | iloveroses |
| 215403 | Johnny  | Appleseed | JohnApple@gmail.com | None         | Oasis Clinic | Lakers248 |        0 |        0 | (403) 555-9898 | Calgary403 |
+--------+---------+----------+---------------------+--------------+-------------+----------+----------+----------+----------------+-----------+
7 rows in set (0.00 sec)
```

Note the last tuple in doctor: the newly added Johnny Appleseed

U.3: NULL protection via required form fields



Attempting to press submit with a field left blank
~(note that ALL text fields in all forms in our system have this protection built in to avoid null values)

U.4: Updated schedule

```
mysql> select * from doctorschedule;
+--------+-------+-------------+-----------------+-----------+
| ID     | Hours | VacatonDays | Days            | Username  |
+--------+-------+-------------+-----------------+-----------+
|     12 |     8 | Mondays     | T-F             | koe       |
|    123 |    18 | july 22     | Monday-Tuesday  | userin    |
| 215403 |    22 | December 25 | Thursday-Sunday | Lakers248 |
+--------+-------+-------------+-----------------+-----------+
3 rows in set (0.00 sec)
```

U.5: Updated Schedule (overwritten existing)

```
mysql> select * from doctorschedule;
+--------+-------+---------------------------+----------------+-----------+
| ID     | Hours | VacatonDays               | Days           | Username  |
+--------+-------+---------------------------+----------------+-----------+
|     12 |     8 | Mondays                   | T-F            | koe       |
|    123 |    18 | july 22                   | Monday-Tuesday | userin    |
| 215403 |     7 | every last month's friday | Monday-Friday  | Lakers248 |
+--------+-------+---------------------------+----------------+-----------+
3 rows in set (0.00 sec)
```

Note the change for username Lakers248 (Johnny Appleseed)

U.6: Addition of a prescription to prescribes

```
mysql> select * from prescribes;
+--------+-------------+-----------+------------+-----------+----------------+----------------+
| ID     | Name        | Username  | Prescribed | PatientID | DoctorNotes    | Dosage         |
+--------+-------------+-----------+------------+-----------+----------------+----------------+
|    123 | Klonopin    | a         | 09/06/2017 | 11111     | Take at night  | 890mg          |
|    123 | menthol     | a         | 1/1/2000   | 11111     | none           | as needed      |
|    123 | Propranolol | a         | 6/12/2018  | 11111     | Avoid Driving  | 100mg 2x daily |
|    123 | Xanax       | a         | 11/07/2018 | 11111     | Take at night  | 400mg          |
| 215403 | Pepto       | Lakers248 | 6/17/2022  | 11111     | Take in morning | 400mg daily    |
+--------+-------------+-----------+------------+-----------+----------------+----------------+
5 rows in set (0.00 sec)
```

Note the addition of the pepto prescription we created as an example

U.7: Requested Drug

```
mysql> select * from drugs;
+--------------+--------------------------------------+-------------+
| Company      | SideEffects                          | Name        |
+--------------+--------------------------------------+-------------+
| 123          | 123                                  | 123         |
| NULL         | NULL                                 | Adderall    |
| Balmoral     | None                                 | Ativan      |
| BigPharma    | Cough and runny nose                 | Clenbuterol |
| NULL         | NULL                                 | Clorazepam  |
| BigPharma    | Cough and runny nose                 | Hydroxyzine |
| BigPharma    | Cough and runny nose                 | Ibuprofen   |
| BigPharma    | Cough and runny nose                 | Klonopin    |
| naturalPharm | none                                 | menthol     |
| PharmMD      | Dizziness, diarrhea, confusion, etc. | Metranolol  |
| BigPharma    | Cough and runny nose                 | Pepto       |
| Drug Inc.    | Drowsiness, Irritability             | Propranolol |
| BigPharma    | Cough and runny nose                 | Tylenol     |
| BigPharma    | Cough and runny nose                 | Xanax       |
+--------------+--------------------------------------+-------------+
14 rows in set (0.00 sec)
```

Note that this is before the pharmacist has updated the values for this requested drug

U.8: Nurse Ri Lu before and after assignment(s)

U.8.1: Nurse Ri Lu before any assignment

```
mysql> select * from nurse;
+-----+-------+---------+-------------+---------------+----------+------------+
| ID  | Fname | Lname   | ClinicName  | Specialty     | Username | Password   |
+-----+-------+---------+-------------+---------------+----------+------------+
|   3 | Ri    | Lu      | Oasis Clinic | Many of them | pas      | word       |
| 123 | Jaa   | Daniels | Oasis Clinic | Special Nurse | userty   | passwordty |
+-----+-------+---------+-------------+---------------+----------+------------+
2 rows in set (0.00 sec)
```

U.8.2: After assigning to XClinic

```
mysql> select * from nurse;
+-----+-------+---------+--------------+---------------+----------+------------+
| ID  | Fname | Lname   | ClinicName   | Specialty     | Username | Password   |
+-----+-------+---------+--------------+---------------+----------+------------+
|   3 | Ri    | Lu      | XClinic      | Many of them  | pas      | word       |
| 123 | Jaa   | Daniels | Oasis Clinic | Special Nurse | userty   | passwordty |
+-----+-------+---------+--------------+---------------+----------+------------+
2 rows in set (0.00 sec)
```

U.8.3: Assigning to XClinic again:

```
mysql> select * from nurse;
+-----+-------+---------+--------------+---------------+----------+------------+
| ID  | Fname | Lname   | ClinicName   | Specialty     | Username | Password   |
+-----+-------+---------+--------------+---------------+----------+------------+
|   3 | Ri    | Lu      | XClinic      | Many of them  | pas      | word       |
| 123 | Jaa   | Daniels | Oasis Clinic | Special Nurse | userty   | passwordty |
+-----+-------+---------+--------------+---------------+----------+------------+
2 rows in set (0.00 sec)
```

U.9: Inserting patient history

U.9.1: all pre-existing patient_medicalhis tuples (ie. before)

```
mysql> select * from patient_medicalhis;
+-------+-----------------------+
| ID    | MedicalHistory        |
+-------+-----------------------+
| 11111 | Heart AttackBad Cough |
+-------+-----------------------+
1 row in set (0.00 sec)
```

U.9.2: Added medical history

```
mysql> select * from patient_medicalhis;
+--------+-----------------------+
| ID     | MedicalHistory        |
+--------+-----------------------+
|  11111 | Heart AttackBad Cough |
| 462380 | Vertigo and cough     |
+--------+-----------------------+
2 rows in set (0.00 sec)
```

U.9.3: Concatenated updated medical history

```
mysql> select * from patient_medicalhis;
+--------+-----------------------------------------+
| ID     | MedicalHistory                          |
+--------+-----------------------------------------+
|  11111 | Heart AttackBad Cough                   |
| 462380 | Wheezing and dry mouth , Vertigo and cough |
+--------+-----------------------------------------+
2 rows in set (0.00 sec)
```

U.10: Inserting patient conditions

U.10.1: Patient conditions before insert

```
mysql> select * from patient_conditions;
+-------+-----------------------------------------------------------+
| ID    | Conditions                                                |
+-------+-----------------------------------------------------------+
| 11111 | Routine Spinal Tap (2009) , Aertrial Bypass Surgery(08/2008) |
+-------+-----------------------------------------------------------+
1 row in set (0.00 sec)
```

U.10.2: Conditions after insert

```
mysql> select * from patient_conditions;
+--------+-----------------------------------------------------------+
| ID     | Conditions                                                |
+--------+-----------------------------------------------------------+
|  11111 | Routine Spinal Tap (2009) , Aertrial Bypass Surgery(08/2008) |
| 462380 | Major surgery                                             |
+--------+-----------------------------------------------------------+
2 rows in set (0.00 sec)
```

U.10.3: Conditions after another insert (note the concatenation)

```
mysql> select * from patient_conditions;
+--------+-----------------------------------------------------------+
| ID     | Conditions                                                |
+--------+-----------------------------------------------------------+
|  11111 | Routine Spinal Tap (2009) , Aertrial Bypass Surgery(08/2008) |
| 462380 | Clubfoot , Major surgery                                  |
+--------+-----------------------------------------------------------+
2 rows in set (0.00 sec)
```

U.11: Pharmacist after registration

```
mysql> select * from pharmacist;
+---------+------------+-------+-----------+------------+----------+
| ID      | Fname      | Lname | Supply    | Username   | Password |
+---------+------------+-------+-----------+------------+----------+
|      56 | Jordan     | Swiss | none      | cat_nyr    | eric     |
|    1234 | John       | Doe   | SupplyOne | JohnDoe123 | mypass   |
|   12345 | jar        | lan   | SupplyO i | jr_l       | 123ref   |
| 9853107 | Pharmacist | Pharm | none      | pharm_md   | pharmacy |
+---------+------------+-------+-----------+------------+----------+
4 rows in set (0.00 sec)
```

U.12: Drugs after adding highlighted drug:

```
mysql> select * from drugs;
+--------------+--------------------------------------+------------+
| Company      | SideEffects                          | Name       |
+--------------+--------------------------------------+------------+
| 123          | 123                                  | 123        |
| NULL         | NULL                                 | Adderall   |
| Balmoral     | None                                 | Ativan     |
| BigPharma    | Cough and runny nose                 | Clenbuterol |
| NULL         | NULL                                 | Clorazepam |
| BigPharma    | Cough and runny nose                 | Hydroxyzine |
| BigPharma    | Cough and runny nose                 | Ibuprofen  |
| BigPharma    | Cough and runny nose                 | Klonopin   |
| naturalPharm | none                                 | menthol    |
| PharmMD      | Dizziness, diarrhea, confusion, etc. | Metranolol |
| LundyCo.     | blurred vision, low bp               | minoxidil  |
| BigPharma    | Cough and runny nose                 | Pepto      |
| Drug Inc.    | Drowsiness, Irritability             | Propranolol |
| BigPharma    | Cough and runny nose                 | Tylenol    |
| BigPharma    | Cough and runny nose                 | Xanax      |
+--------------+--------------------------------------+------------+
15 rows in set (0.00 sec)
```

U.13: No duplicate minoxidil after attempt

```
mysql> select * from drugs;
+--------------+--------------------------------------+-------------+
| Company      | SideEffects                          | Name        |
+--------------+--------------------------------------+-------------+
| 123          | 123                                  | 123         |
| NULL         | NULL                                 | Adderall    |
| Balmoral     | None                                 | Ativan      |
| BigPharma    | Cough and runny nose                 | Clenbuterol |
| NULL         | NULL                                 | Clorazepam  |
| BigPharma    | Cough and runny nose                 | Hydroxyzine |
| BigPharma    | Cough and runny nose                 | Ibuprofen   |
| BigPharma    | Cough and runny nose                 | Klonopin    |
| naturalPharm | none                                 | menthol     |
| PharmMD      | Dizziness, diarrhea, confusion, etc. | Metranolol  |
| LundyCo.     | blurred vision, low bp               | minoxidil   |
| BigPharma    | Cough and runny nose                 | Pepto       |
| Drug Inc.    | Drowsiness, Irritability             | Propranolol |
| BigPharma    | Cough and runny nose                 | Tylenol     |
| BigPharma    | Cough and runny nose                 | Xanax       |
+--------------+--------------------------------------+-------------+
15 rows in set (0.00 sec)
```

U.14: All drugs, note the drugs with NULL attributes are drugs that doctors have requested recently that no pharmacist has filled yet

```
mysql> select * from drugs;
+--------------+--------------------------------------+-------------+
| Company      | SideEffects                          | Name        |
+--------------+--------------------------------------+-------------+
| 123          | 123                                  | 123         |
| NULL         | NULL                                 | Adderall    |
| Balmoral     | None                                 | Ativan      |
| BigPharma    | Cough and runny nose                 | Clenbuterol |
| NULL         | NULL                                 | Clorazepam  |
| BigPharma    | Cough and runny nose                 | Hydroxyzine |
| BigPharma    | Cough and runny nose                 | Ibuprofen   |
| BigPharma    | Cough and runny nose                 | Klonopin    |
| naturalPharm | none                                 | menthol     |
| PharmMD      | Dizziness, diarrhea, confusion, etc. | Metranolol  |
| LundyCo.     | blurred vision, low bp               | minoxidil   |
| BigPharma    | Cough and runny nose                 | Pepto       |
| Drug Inc.    | Drowsiness, Irritability             | Propranolol |
| BigPharma    | Cough and runny nose                 | Tylenol     |
| BigPharma    | Cough and runny nose                 | Xanax       |
+--------------+--------------------------------------+-------------+
15 rows in set (0.00 sec)
```

U.15: Change after fulfilling adderall request

```
mysql> select * from drugs;
+--------------+--------------------------------------+-------------+
| Company      | SideEffects                          | Name        |
+--------------+--------------------------------------+-------------+
| 123          | 123                                  | 123         |
| BigPharma    | Confusion, night sweats              | Adderall    |
| Balmoral     | None                                 | Ativan      |
| BigPharma    | Cough and runny nose                 | Clenbuterol |
| NULL         | NULL                                 | Clorazepam  |
| BigPharma    | Cough and runny nose                 | Hydroxyzine |
| BigPharma    | Cough and runny nose                 | Ibuprofen   |
| BigPharma    | Cough and runny nose                 | Klonopin    |
| naturalPharm | none                                 | menthol     |
| PharmMD      | Dizziness, diarrhea, confusion, etc. | Metranolol  |
| LundyCo.     | blurred vision, low bp               | minoxidil   |
| BigPharma    | Cough and runny nose                 | Pepto       |
| Drug Inc.    | Drowsiness, Irritability             | Propranolol |
| BigPharma    | Cough and runny nose                 | Tylenol     |
| BigPharma    | Cough and runny nose                 | Xanax       |
+--------------+--------------------------------------+-------------+
15 rows in set (0.00 sec)
```

U.16: Submitting a new nurse to the system

```
mysql> select * from nurse;
+-----------+-------+---------+-------------+--------------------+----------+------------+
| ID        | Fname | Lname   | ClinicName  | Specialty          | Username | Password   |
+-----------+-------+---------+-------------+--------------------+----------+------------+
|         3 | Ri    | Lu      | XClinic     | Many of them       | pas      | word       |
|       123 | Jaa   | Daniels | Oasis Clinic | Special Nurse      | userty   | passwordty |
| 875587857 | User  | Manual  | Oasis Clinic | Emergency medicine | Example  | password   |
+-----------+-------+---------+-------------+--------------------+----------+------------+
3 rows in set (0.01 sec)
```

U.17: Changing nurse schedule

```
mysql> select * from nurseschedule;
+-----------+-------+----------------------+------------------+----------+
| ID        | Hours | VacatonDays          | Days             | Username |
+-----------+-------+----------------------+------------------+----------+
|         3 |    10 | may 12               | Tuesday-Saturday | pas      |
| 875587857 |    12 | Easter and Christmas | Monday-Friday    | Example  |
+-----------+-------+----------------------+------------------+----------+
2 rows in set (0.00 sec)
```

Compare to appendix u.1 for nurse schedule

U.18: Updated nurse schedule

```
mysql> select * from nurseschedule;
+-----------+-------+-------------+-------------------+----------+
| ID        | Hours | VacatonDays | Days              | Username |
+-----------+-------+-------------+-------------------+----------+
|         3 |    10 | may 12      | Tuesday-Saturday  | pas      |
| 875587857 |     4 | none        | Monday-Thursday   | Example  |
+-----------+-------+-------------+-------------------+----------+
2 rows in set (0.00 sec)
```

U.19: assigning nurse to XClinic

```
mysql> select * from nurse;
+-----------+-------+---------+--------------+--------------------+----------+------------+
| ID        | Fname | Lname   | ClinicName   | Specialty          | Username | Password   |
+-----------+-------+---------+--------------+--------------------+----------+------------+
|         3 | Ri    | Lu      | XClinic      | Many of them       | pas      | word       |
|       123 | Jaa   | Daniels | Oasis Clinic | Special Nurse      | userty   | passwordty |
| 875587857 | User  | Manual  | XClinic      | Emergency medicine | Example  | password   |
+-----------+-------+---------+--------------+--------------------+----------+------------+
3 rows in set (0.00 sec)
```

U20: Assigning back to Oasis via link on view clinics

```
mysql> select * from nurse;
+-----------+-------+---------+--------------+--------------------+----------+------------+
| ID        | Fname | Lname   | ClinicName   | Specialty          | Username | Password   |
+-----------+-------+---------+--------------+--------------------+----------+------------+
|         3 | Ri    | Lu      | XClinic      | Many of them       | pas      | word       |
|       123 | Jaa   | Daniels | Oasis Clinic | Special Nurse      | userty   | passwordty |
| 875587857 | User  | Manual  | Oasis Clinic | Emergency medicine | Example  | password   |
+-----------+-------+---------+--------------+--------------------+----------+------------+
3 rows in set (0.00 sec)
```

U21: HAS table indicating appointments with pre-existing values

```
+----------+---------+-----------+-------------------+-------------+---------+-----------+----------------+
| DoctorID | NurseID | PatientID | AppointmentNumber | PatientUser | DocUser | NurseUser | TimeDate       |
+----------+---------+-----------+-------------------+-------------+---------+-----------+----------------+
|       32 |     123 |     11111 |                 2 | user        | fna@    | userty    | 3:45,3/10/2022 |
+----------+---------+-----------+-------------------+-------------+---------+-----------+----------------+
1 row in set (0.00 sec)
```

U22: Has & Appointment table indicating that patient Pat Zero(id 11111) has an active appointment

```
mysql> select * from appointment;
+-------+-------------------+-----------------+
| ID    | AppointmentNumber | AppointmentType |
+-------+-------------------+-----------------+
| 11111 |                 2 | Checkup         |
+-------+-------------------+-----------------+
1 row in set (0.00 sec)

mysql> select * from has;
+----------+---------+-----------+-------------------+-------------+---------+-----------+----------------+
| DoctorID | NurseID | PatientID | AppointmentNumber | PatientUser | DocUser | NurseUser | TimeDate       |
+----------+---------+-----------+-------------------+-------------+---------+-----------+----------------+
|       32 |     123 |     11111 |                 2 | user        | fna@    | userty    | 3:45,3/10/2022 |
+----------+---------+-----------+-------------------+-------------+---------+-----------+----------------+
1 row in set (0.00 sec)
```

Note that this has instance is connected via the foreign key appointment number to appointment, and they both are connected to Pat Zero patient via the ID (11111).

U23: All doctors from Oasis Clinic

```
mysql> select * from doctor where clinicname='Oasis Clinic';
+--------+---------+----------+--------------------+--------------+-------------+----------+---------+---------+----------------+------------+
| ID     | Fname   | Lname    | Email              | Specialty    | ClinicName  | Username | Spe_Flag | Fam_Flag | Phone          | Password   |
+--------+---------+----------+--------------------+--------------+-------------+----------+---------+---------+----------------+------------+
|     12 | joe     | joe      | joe                | joe          | Oasis Clinic | koe      |        1 |        0 | no             | nope       |
|     32 | Fir     | Nam      | jr4@yahoo.com      | Specialist   | Oasis Clinic | fna@     |        1 |        0 | 9086676        | 12345tyu   |
|    123 | a       | a        | a                  | Family Doctor | Oasis Clinic | a        |        1 |        0 | a              | a          |
|    123 | FName   | LName    | emailin            | specialty    | Oasis Clinic | userin   |        1 |        0 | phonein        | pswdin     |
|    453 | Ros     | Ln       | tl@tip.com         | Family Doctor | Oasis Clinic | r_ln     |        0 |        1 | 908-667-3218   | pass       |
|  89765 | Michael | Rose     | dr_mrose@gmail.com | Specialist   | Oasis Clinic | mrose4567 |        1 |        0 | 403-999-9999   | iloveroses |
| 215403 | Johnny  | Appleseed | JohnApple@gmail.com | None         | Oasis Clinic | Lakers248 |        0 |        0 | (403) 555-9898 | Calgary403 |
+--------+---------+----------+--------------------+--------------+-------------+----------+---------+---------+----------------+------------+
7 rows in set (0.00 sec)
```

Note that this matches the dropdown menu option in our example

U24: HAS and Appointment tables following our example insertion (our example booking)

```
mysql> select * from appointment;
+--------+-------------------+-----------------+
| ID     | AppointmentNumber | AppointmentType |
+--------+-------------------+-----------------+
|  11111 |                 2 | Checkup         |
| 462380 |                 3 | checkup         |
+--------+-------------------+-----------------+
2 rows in set (0.00 sec)

mysql> select * from has;
+----------+---------+-----------+-------------------+-------------+-----------+-----------+-----------------+
| DoctorID | NurseID | PatientID | AppointmentNumber | PatientUser | DocUser   | NurseUser | TimeDate        |
+----------+---------+-----------+-------------------+-------------+-----------+-----------+-----------------+
|       32 |     123 |     11111 |                 2 | user        | fna@      | userty    | 3:45,3/10/2022  |
|   215403 |     123 |    462380 |                 3 | Chris_Breaux | Lakers248 | userty    | 12:45,5/15/2022 |
+----------+---------+-----------+-------------------+-------------+-----------+-----------+-----------------+
2 rows in set (0.00 sec)
```

U24: Prescribes table for all prescriptions following our user manual samples

```
mysql> select * from prescribes;
+--------+-------------+-----------+------------+-----------+-----------------------------+----------------+
| ID     | Name        | Username  | Prescribed | PatientID | DoctorNotes                 | Dosage         |
+--------+-------------+-----------+------------+-----------+-----------------------------+----------------+
|    123 | Klonopin    | a         | 09/06/2017 | 11111     | Take at night               | 890mg          |
|    123 | menthol     | a         | 1/1/2000   | 11111     | none                        | as needed      |
|    123 | Propranolol | a         | 6/12/2018  | 11111     | Avoid Driving               | 100mg 2x daily |
|    123 | Xanax       | a         | 11/07/2018 | 11111     | Take at night               | 400mg          |
| 215403 | Ativan      | Lakers248 | 12/4/2022  | 11111     | Call 911 if signs of stroke | 100mg biweekly |
| 215403 | Pepto       | Lakers248 | 6/17/2022  | 11111     | Take in morning             | 400mg daily    |
+--------+-------------+-----------+------------+-----------+-----------------------------+----------------+
6 rows in set (0.01 sec)
```

Note this is all prescriptions, and patients can only view the prescriptions mapped to their ID only

U25: Appointment and Has Tables before cancelling appointment

```
mysql> select * from appointment;
+--------+-------------------+-----------------+
| ID     | AppointmentNumber | AppointmentType |
+--------+-------------------+-----------------+
|  11111 |                 2 | Checkup         |
| 462380 |                 3 | checkup         |
+--------+-------------------+-----------------+
2 rows in set (0.00 sec)

mysql> select * from has;
+----------+---------+-----------+-------------------+-------------+-----------+-----------+-----------------+
| DoctorID | NurseID | PatientID | AppointmentNumber | PatientUser | DocUser   | NurseUser | TimeDate        |
+----------+---------+-----------+-------------------+-------------+-----------+-----------+-----------------+
|       32 |     123 |     11111 |                 2 | user        | fna@      | userty    | 3:45,3/10/2022  |
|   215403 |     123 |    462380 |                 3 | Chris_Breaux | Lakers248 | userty    | 12:45,5/15/2022 |
+----------+---------+-----------+-------------------+-------------+-----------+-----------+-----------------+
2 rows in set (0.00 sec)
```

Note that Pat Zero has ID=11111

U26: U25 after patient 11111(Pat Zero) cancels his appointment

```
mysql> select * from appointment;
+--------+-------------------+-----------------+
| ID     | AppointmentNumber | AppointmentType |
+--------+-------------------+-----------------+
| 462380 |                 3 | checkup         |
+--------+-------------------+-----------------+
1 row in set (0.00 sec)

mysql> select * from has;
+----------+---------+-----------+-------------------+-------------+-----------+-----------+-----------------+
| DoctorID | NurseID | PatientID | AppointmentNumber | PatientUser | DocUser   | NurseUser | TimeDate        |
+----------+---------+-----------+-------------------+-------------+-----------+-----------+-----------------+
|   215403 |     123 |    462380 |                 3 | Chris_Breaux | Lakers248 | userty    | 12:45,5/15/2022 |
+----------+---------+-----------+-------------------+-------------+-----------+-----------+-----------------+
1 row in set (0.00 sec)
```

Compared to U25 (before), this is the aftermath of cancelling an appointment

Appendix U27: Environment

This project was coded entirely using HTML, CSS, Java, JSP, Java Servlets and mySQL.
The code was created in Eclipse with Dynamic Web Project plugins for Java installed.
The project was run on a localhost (port 8080) server from Eclipse to Microsoft Edge.
The runtime server used was Apache Tomcat v9.0.
All SQL functionality was implemented using mySQL and the JDBC Connection JAR.

Credit & References

This report was entirely written by group 43 members Jared Lundy, Jordan Lundy and Yuhao Guang. The group members communicated personally as a group via the team Discord channel.

The class textbook, *Fundamentals of Database Systems, Seventh Edition (Authors: Ramez Elmasri & Shamkant B. Navathe, 2016)* was referred to in the design process (ERD Diagram, UML Diagram, Relational Model, Mapping ERD to Relational model), and the theory of databases presented in this text was used throughout.

Dr. Jalal Kawah's CPSC 471 class notes and the CPSC 471 Tutorial Notes (both accessed via D2L) were also referred to.